IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION

AMERICAN PATENTS LLC, Plaintiff, v. ANALOG DEVICES, INC., CYPRESS SEMICONDUCTOR CORPORATION, MARVELL INTERNATIONAL, LTD., MEDIATEK INC., and MEDIATEK USA INC., Defendants.	CIVIL ACTION NO. 6:18-CV-356 ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT **JURY TRIAL DEMANDED**

ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff American Patents LLC ("American Patents" or "Plaintiff") files this original complaint against Defendants Analog Devices, Inc., Cypress Semiconductor Corporation, Marvell International, Ltd., MediaTek Inc., and MediaTek USA Inc. (collectively "Defendants"), alleging, based on its own knowledge as to itself and its own actions and based on information and belief as to all other matters, as follows:

PARTIES

1. American Patents is a limited liability company formed under the laws of the State of Texas, with its principal place of business at 2325 Oak Alley, Tyler, Texas, 75703.

2. Analog Devices, Inc. ("Analog") is a corporation organized and existing under the laws of the state of Massachusetts. It can be served via its registered agent: CT Corp System, 1999 Bryan Street, Suite 900, Dallas, TX 75201.

3. Analog is one of the world's largest manufacturers of integrated circuits.

4. Cypress Semiconductor Corporation ("Cypress") is a corporation organized under the laws of the state of Delaware. Cypress can be served with process by serving its registered agent: Corporation Service Company d/b/a CSC-Lawyers Incorporating Service Company, 211 E. 7th Street, Suite 620, Austin, Texas 78701.

5. Cypress is one of the world's largest manufacturers of integrated circuits.

6. Marvell International, Ltd. ("Marvell") is a company organized under the laws of Bermuda. Marvell has an office at Canon's Court, 22 Victoria Street, Hamilton, HM 12, Bermuda.

7. Marvell is one of the world's largest manufacturers of integrated circuits.

8. MediaTek Inc. is a company incorporated under the laws of Taiwan, having an address of No. 1, Dusing Road 1, Hsinchu Science Park, Hsinchu City 30078, Taiwan.

9. MediaTek USA Inc. is a company incorporated under the laws of the State of Delaware and having an established place of business at 5914 W. Courtyard Drive, Austin, Texas 78730. MediaTek USA is registered to conduct business in Texas and may be served through its registered agent, CT Corporation System, 1999 Bryan Street, Suite 900, Dallas, Texas 75201-3136.

10. The Defendants identified in paragraphs 8 and 9 above (collectively, "MediaTek") are companies which together comprise one of the world's largest manufacturers of integrated circuits.

11. The MediaTek defendants named above are part of the same corporate structure and distribution chain for the making, importing, offering to sell, selling, and/or using of the accused devices in the United States, including in the State of Texas generally and this judicial district in particular.

12. The MediaTek defendants named above share the same management, common ownership, advertising platforms, facilities, distribution chains and platforms, and accused product lines and products involving related technologies.

13. Thus, the MediaTek defendants named above operate as a unitary business venture and are jointly and severally liable for the acts of patent infringement alleged herein.

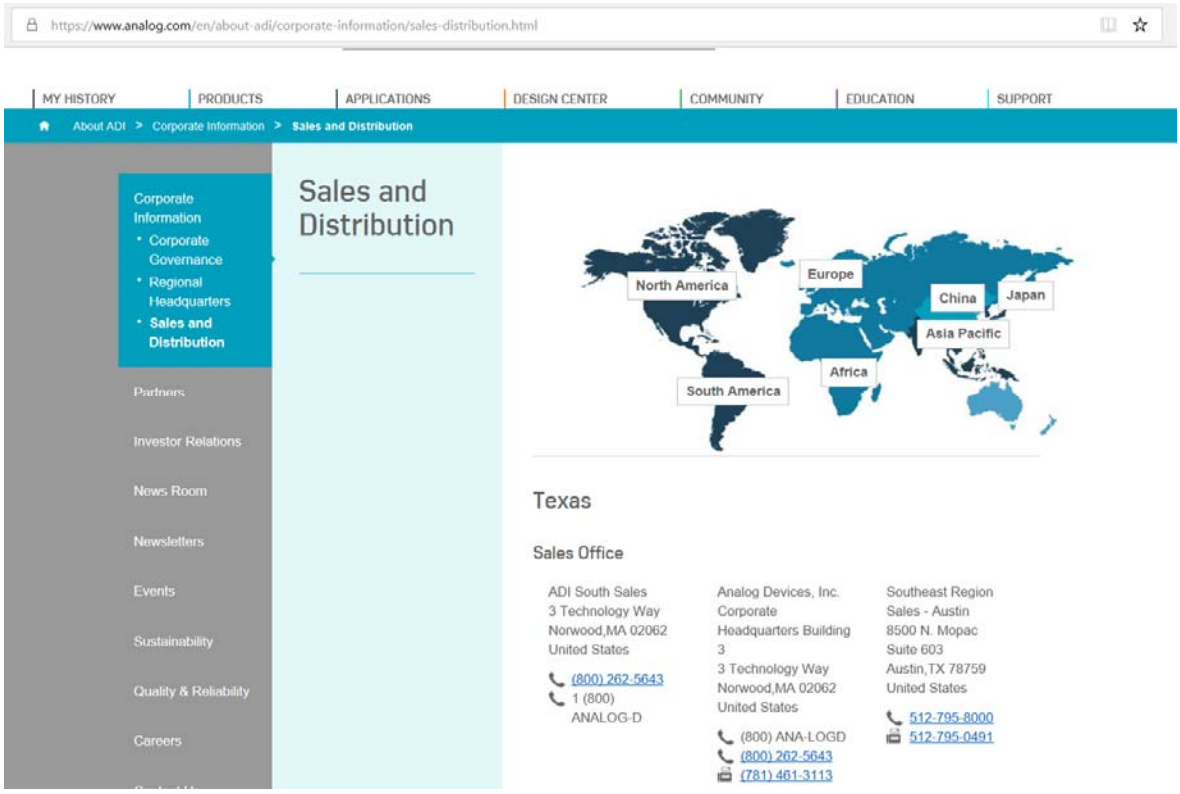
14. The parties to this action are properly joined under 35 U.S.C. § 299 because the right to relief asserted against Defendants jointly and severally arises out of the same series of transactions or occurrences relating to the making and using of the same products or processes, including products using the processors and related processes based on common ARM architectures. Additionally, questions of fact common to all defendants will arise in this action.

JURISDICTION AND VENUE

15. This is an action for infringement of United States patents arising under 35 U.S.C. §§ 271, 281, and 284–85, among others. This Court has subject matter jurisdiction of the action under 28 U.S.C. § 1331 and § 1338(a).

16. This Court has personal jurisdiction over Defendants pursuant to due process and/or the Texas Long Arm Statute because, *inter alia*, (i) Defendants have done and continue to do business in Texas and (ii) Defendants have committed and continue to commit acts of patent infringement in the State of Texas, including making, using, offering to sell, and/or selling accused products in Texas, and/or importing accused products into Texas, including by Internet sales and sales via retail and wholesale stores, inducing others to commit acts of patent infringement in Texas, and/or committing a least a portion of any other infringements alleged herein. In addition, or in the alternative, this Court has personal jurisdiction over Defendants pursuant to Fed. R. Civ. P. 4(k)(2).

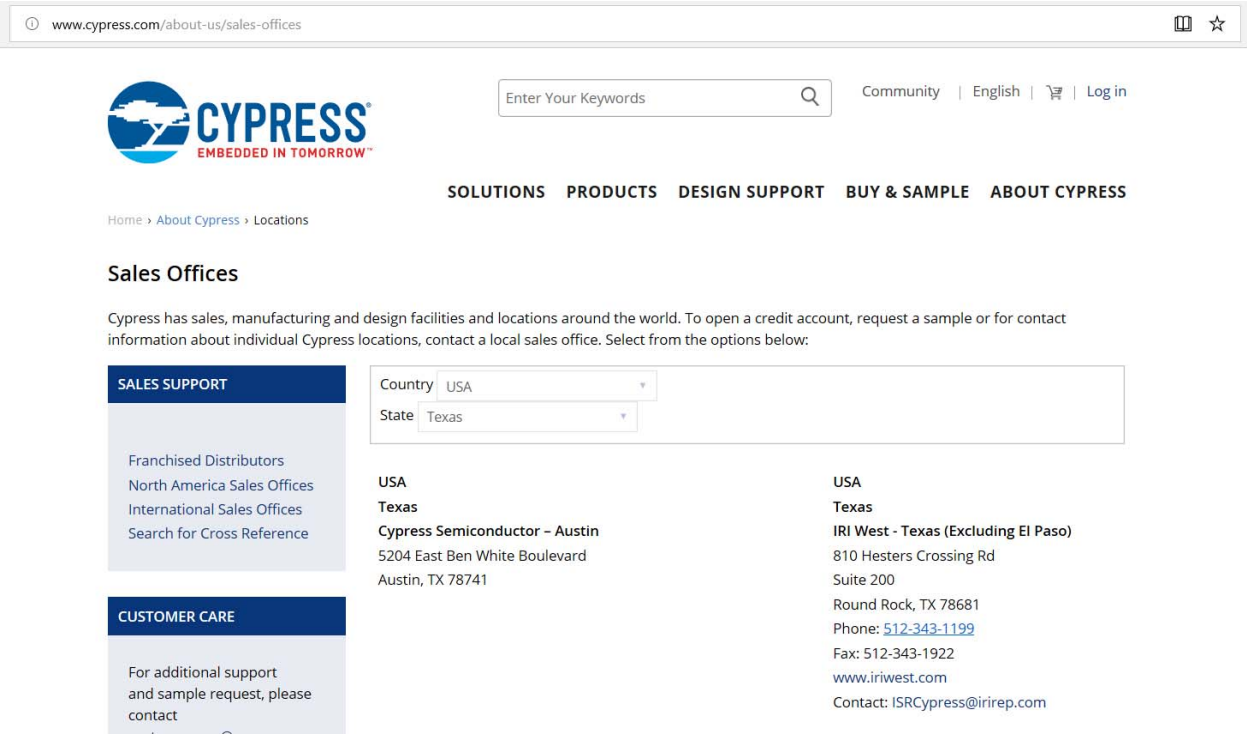
17. Venue is proper in this district pursuant to 28 U.S.C. §§ 1391(b), 1391(c), and 1400(b) because (i) Analog has done and continues to do business in this district; (ii) Analog has committed and continues to commit acts of patent infringement in this district, including making, using, offering to sell, and/or selling accused products in this district, and/or importing accused products into this district, including by internet sales and sales via retail and wholesale stores, and/or inducing others to commit acts of patent infringement in this district; and (iii) Analog has a regular and established place of business in this district at 8500 N. Mopac, Suite 603, Austin, TX 78759, as stated on Analog's website:



https://www.analog.com/en/about-adi/corporate-information/sales-distribution.html

18. Venue is proper in this district pursuant to 28 U.S.C. §§ 1391(b), 1391(c), and 1400(b) because (i) Cypress has done and continues to do business in this district; (ii) Cypress has committed and continues to commit acts of patent infringement in this district, including

making, using, offering to sell, and/or selling accused products in this district, and/or importing accused products into this district, including by internet sales and sales via retail and wholesale stores, and/or inducing others to commit acts of patent infringement in this district; and (iii) Cypress has a regular and established place of business in this district at 5204 East Ben White Boulevard, Austin, Texas 78741 as stated on Cypress's website:

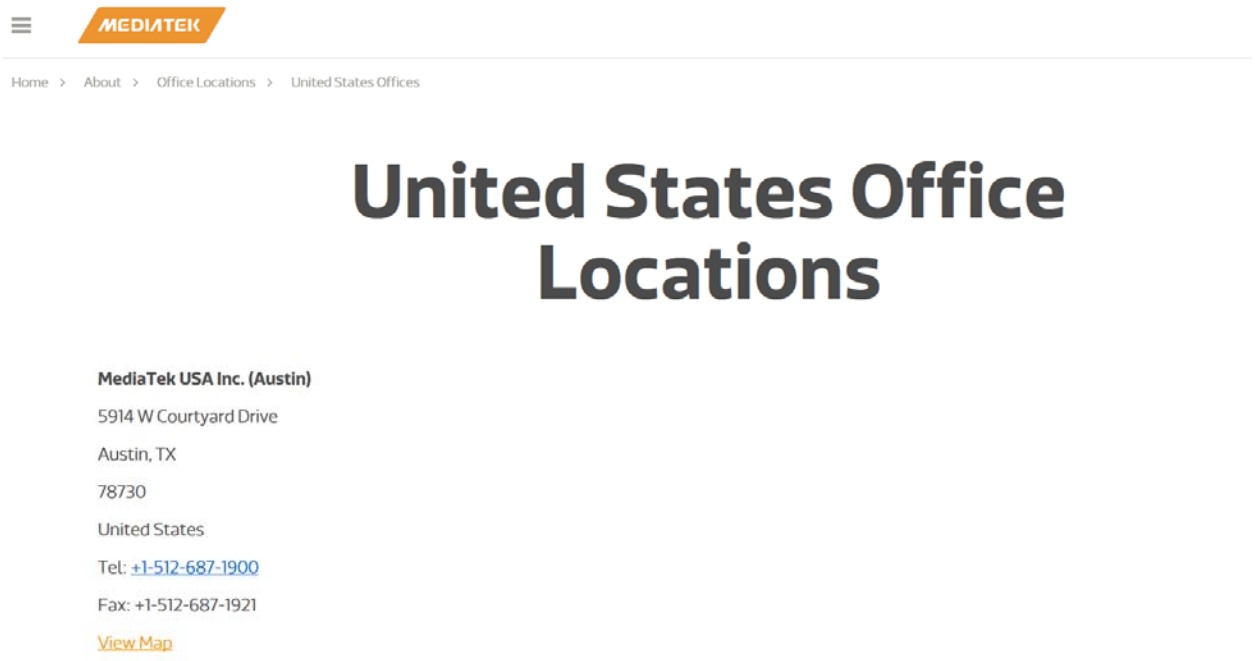


http://www.cypress.com/about-us/sales-offices

19. Venue is proper in this district pursuant to 28 U.S.C. §§ 1391(b), 1391(c), and 1400(b) because (i) Marvell has done and continues to do business in this district; (ii) Marvell has committed and continues to commit acts of patent infringement in this district, including making, using, offering to sell, and/or selling accused products in this district, and/or importing accused products into this district, including by internet sales and sales via retail and wholesale stores, and/or inducing others to commit acts of patent infringement in this district; and (iii) Marvell is a foreign entity.

20. Venue is proper as to Marvell International, Ltd., which is organized under the laws of Bermuda. 28 U.S.C. § 1391(c)(3) provides that "a defendant not resident in the United States may be sued in any judicial district, and the joinder of such a defendant shall be disregarded in determining where the action may be brought with respect to other defendants."

21. Venue is proper in this district pursuant to 28 U.S.C. §§ 1391(b), 1391(c), and 1400(b) because (i) MediaTek has done and continues to do business in this district; (ii) MediaTek has committed and continues to commit acts of patent infringement in this district, including making, using, offering to sell, and/or selling accused products in this district, and/or importing accused products into this district, including by internet sales and sales via retail and wholesale stores, and/or inducing others to commit acts of patent infringement in this district; (iii) MediaTek Inc. is a foreign entity; and (iv) MediaTek USA Inc. has a regular and established place of business in this district at 5914 W. Courtyard Drive, Austin, Texas 78730, as stated on MediaTek's website:



https://www.mediatek.com/about/office-locations/mediatek-usa-offices

22. Venue is proper as to MediaTek Inc., which is organized under the laws of Taiwan. 28 U.S.C. § 1391(c)(3) provides that "a defendant not resident in the United States may be sued in any judicial district, and the joinder of such a defendant shall be disregarded in determining where the action may be brought with respect to other defendants."

BACKGROUND

23. The patents-in-suit generally pertain to hardware circuits for variable-length coding and decoding of media data. The technology disclosed by the patents was developed in the 1990s by employees of Equator Technologies, Inc. including electrical/electronics engineers Richard M. Deeley, Yatin Mundkur, and Dr. Woobin Lee.

24. Prior to the patented technology, coding and decoding of media data with variable-length codes was either done at the software level (which effectively prevented real time encoding or decoding) or in hardware with a specially-designed processor for each specific type of variable-length code (which led to either multiple circuits, increasing the size, complexity, and cost of the decoding hardware, or a single circuit, limiting the hardware to a single type of variable-length code).

25. The inventors solved the problems inherent in these prior solutions by offering a new solution: its VLx processor, a circuit which obtains the speed of a hardware solution with the flexibility of a software solution by allowing flexibility in the processing of incoming bitstreams. The results of their pioneering efforts were recognized by articles in *Electrical Design News* and *Electronic Engineering Times*, and their patented solution is widely used throughout the industry.

COUNT I

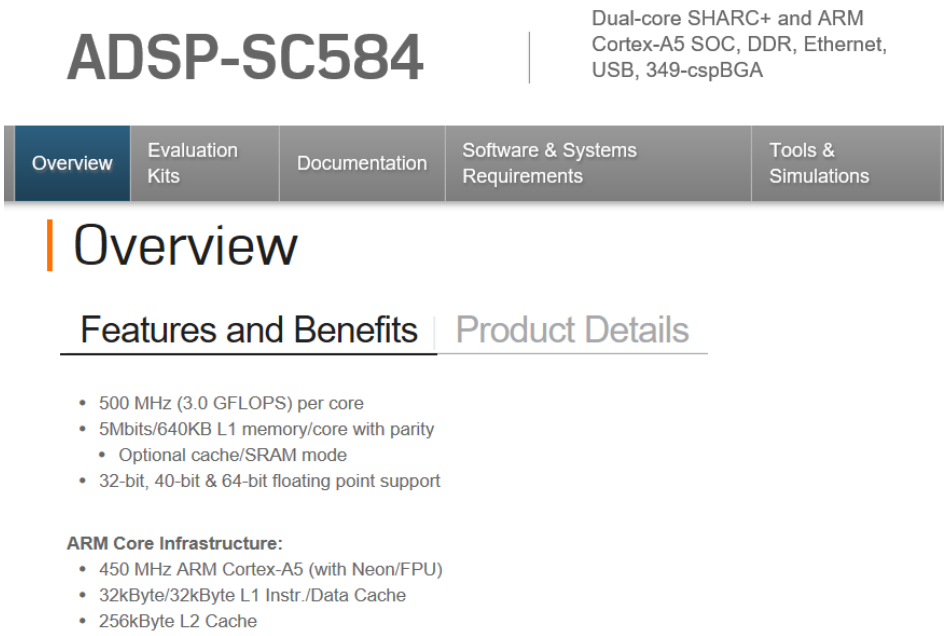
DIRECT INFRINGEMENT OF U.S. PATENT NO. 6,507,293

26. On January 14, 2003, United States Patent No. 6,507,293 ("the '293 Patent") was duly and legally issued by the United States Patent and Trademark Office for an invention entitled "Processing Circuit And Method For Variable-Length Coding And Decoding."

27. American Patents is the owner of the '293 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the '293 Patent against infringers, and to collect damages for all relevant times.

28. Analog made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its ADSP-SC584 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products") [1]:

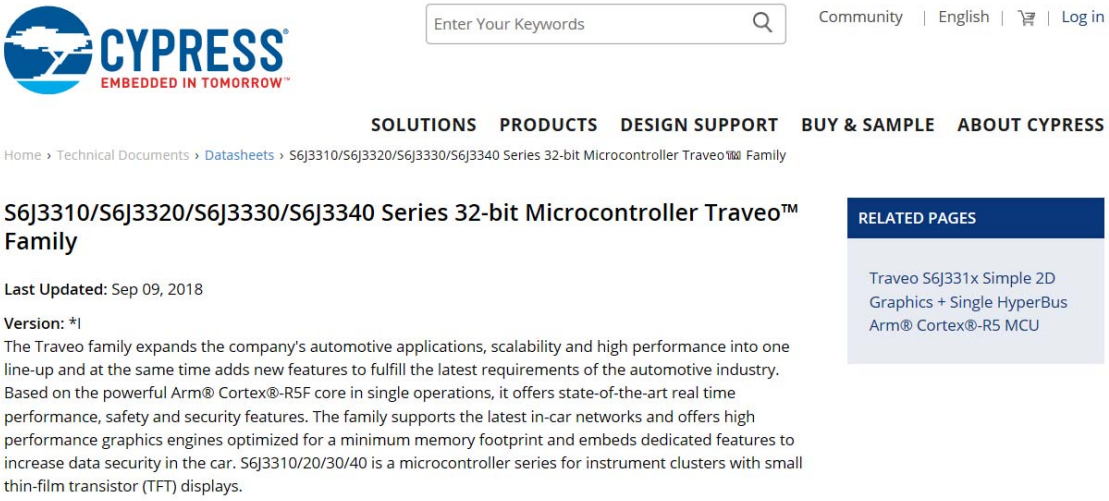
[1] A non-exhaustive list of additional accused products includes the ADSP-SC587 and ADSP-SC589 families of products that include advanced circuits for variable-length coding and decoding of media data.



Source: https://www.analog.com/en/products/adsp-sc584.html#product-overview

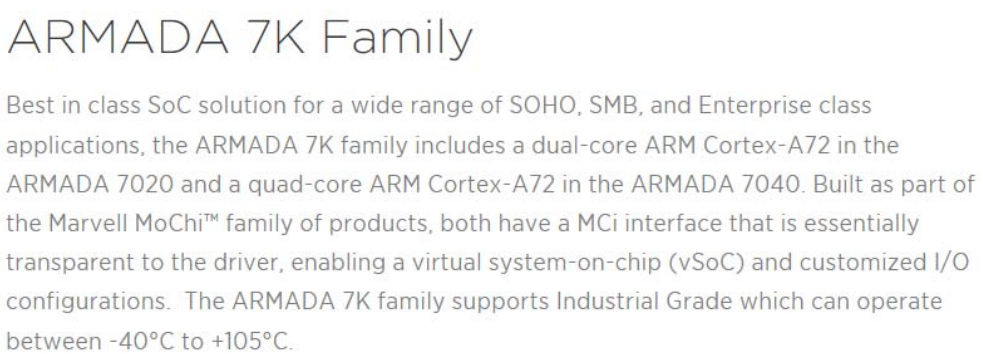
29. Cypress made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Traveo S6J3310 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[2]:

[2] A non-exhaustive list of additional accused products includes the Traveo S6J3320 and S6J3340 families of products that include advanced circuits for variable-length coding and decoding of media data.



Source: http://www.cypress.com/documentation/datasheets/s6j3310s6j3320s6j3330s6j3340-series-32-bit-microcontroller-traveo-family

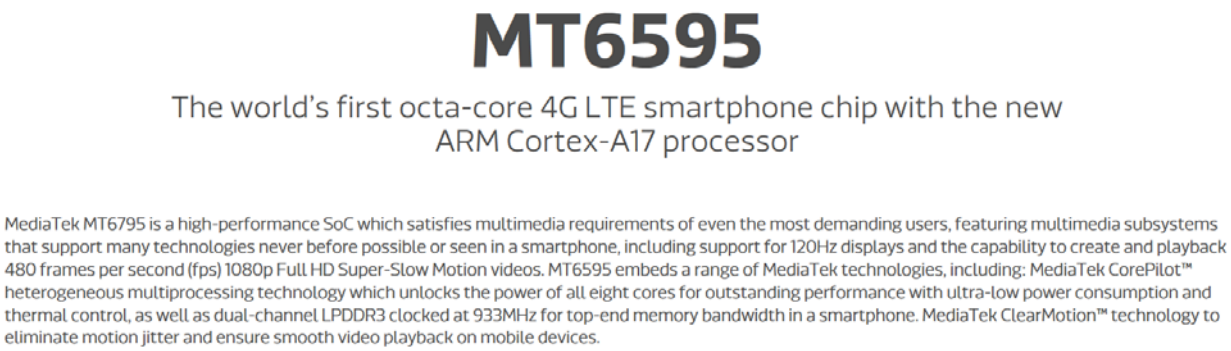
30. Marvell made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its ARMADA 7020 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[3]:



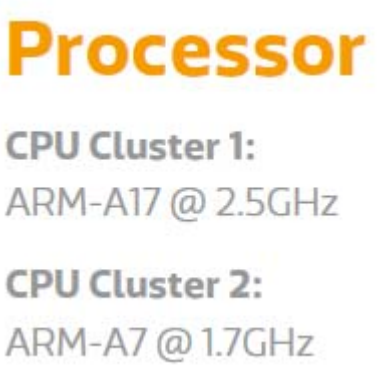
Source: https://www.marvell.com/embedded-processors/armada-70xx/

[3] A non-exhaustive list of additional accused products includes the ARMADA 7040, ARMADA 8020, and ARMADA 8040 families of products that include advanced circuits for variable-length coding and decoding of media data.

31. MediaTek made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its MT6595 and Helio X27 families of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[4]:



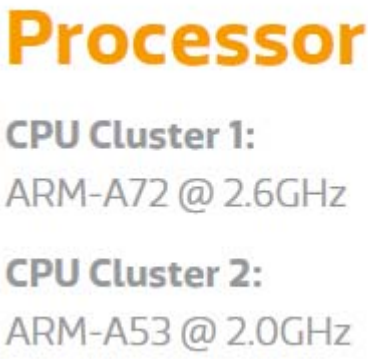
Source: https://www.mediatek.com/products/smartphones/mt6595



Source: https://www.mediatek.com/products/smartphones/mt6595

[4] A non-exhaustive list of additional accused products includes the Helio P60, Helio P70, MT8173, MT8176, and Helio X series (including Helio X20, Helio X23, and Helio X25) families of products that include advanced circuits for variable-length coding and decoding of media data.

Source: https://www.mediatek.com/products/smartphones/mt6797x-helio-x27



Source: https://www.mediatek.com/products/smartphones/mt6797x-helio-x27

32. By doing so, Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least Claim 7 of the '293 Patent. Defendants' infringement in this regard is ongoing.

33. Defendants have infringed the '293 Patent by making, having made, using, importing, providing, supplying, distributing, selling or offering for sale integrated circuits having an advanced function processing block.

34. The accused products include an instruction buffer and memory.

35. The accused products include a getbits processing engine operable to reverse the order of a group of consecutive data bits.

Media Processing Engine

The MPE implements ARM NEON technology, a media and signal processing architecture that adds instructions targeted at audio, video, 3-D graphics, image, and speech processing. Advanced SIMD instructions are available in both ARM and Thumb states.

If the design includes the MPE, the FPU is included.

See the *Cortex-A5 NEON Media Processing Engine Technical Reference Manual*.

Source: ARM Cortex-A5 Technical Reference Manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0433c/DDI0433C_cortex_a5_trm.pdf

The Cortex-A5 processor implements the ARM v7-A architecture profile. This includes:

- the 32-bit ARM instruction set
- the Thumb instruction set, a variable-length instruction set, that has both 16-bit and 32-bit instructions
- execution of 8-bit Java bytecodes
- the ThumbEE instruction set
- the security extensions, TrustZone
- the VFPv4 Floating point architecture and Advanced SIMD extensions, NEON.

Source: ARM Cortex-A5 Technical Reference Manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0433c/DDI0433C_cortex_a5_trm.pdf

ARM architecture

The Cortex-R5 processor implements the ARMv7-R architecture profile that includes the following architecture extensions:

- Advanced *Single Instruction Multiple Data* (SIMD) architecture extension for integer and floating-point vector operations
- *Vector Floating-Point version 3* (VFPv3) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard
- Multiprocessing Extensions for multiprocessing functionality.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

Note

- The Cortex-R5F processor is a Cortex-R5 processor that includes the optional *Floating Point Unit* (FPU) extension.
- In this book, references to the Cortex-R5 processor also apply to the Cortex-R5F processor, unless the context makes it clear that this is not the case.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

Table A4-24 Miscellaneous Advanced SIMD data-processing instructions

Instruction	See
Vector Absolute Difference and Accumulate	*VABA, VABAL* on page A8-819
Vector Absolute Difference	*VABD, VABDL (integer)* on page A8-821 *VABD (floating-point)* on page A8-823
Vector Absolute	*VABS* on page A8-825
Vector Convert between floating-point and fixed point	*VCVT (between floating-point and fixed-point, Advanced SIMD)* on page A8-873
Vector Convert between floating-point and integer	*VCVT (between floating-point and integer, Advanced SIMD)* on page A8-869
Vector Convert between half-precision and single-precision	*VCVT (between half-precision and single-precision, Advanced SIMD)* on page A8-879
Vector Count Leading Sign Bits	*VCLS* on page A8-859
Vector Count Leading Zeros	*VCLZ* on page A8-863
Vector Count Set Bits	*VCNT* on page A8-867
Vector Duplicate scalar	*VDUP (scalar)* on page A8-885
Vector Extract	*VEXT* on page A8-891
Vector Move and Narrow	*VMOVN* on page A8-953
Vector Move Long	*VMOVL* on page A8-951
Vector Maximum, Minimum	*VMAX, VMIN (integer)* on page A8-927 *VMAX, VMIN (floating-point)* on page A8-929
Vector Negate	*VNEG* on page A8-969
Vector Pairwise Maximum, Minimum	*VPMAX, VPMIN (integer)* on page A8-987 *VPMAX, VPMIN (floating-point)* on page A8-989
Vector Reciprocal Estimate	*VRECPE* on page A8-1025
Vector Reciprocal Step	*VRECPS* on page A8-1027
Vector Reciprocal Square Root Estimate	*VRSQRTE* on page A8-1039
Vector Reciprocal Square Root Step	*VRSQRTS* on page A8-1041
Vector Reverse	*VREV16, VREV32, VREV64* on page A8-1029
Vector Saturating Absolute	*VQABS* on page A8-995
Vector Saturating Move and Narrow	*VQMOVN, VQMOVUN* on page A8-1005
Vector Saturating Negate	*VQNEG* on page A8-1007
Vector Swap	*VSWP* on page A8-1093
Vector Table Lookup	*VTBL, VTBX* on page A8-1095

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

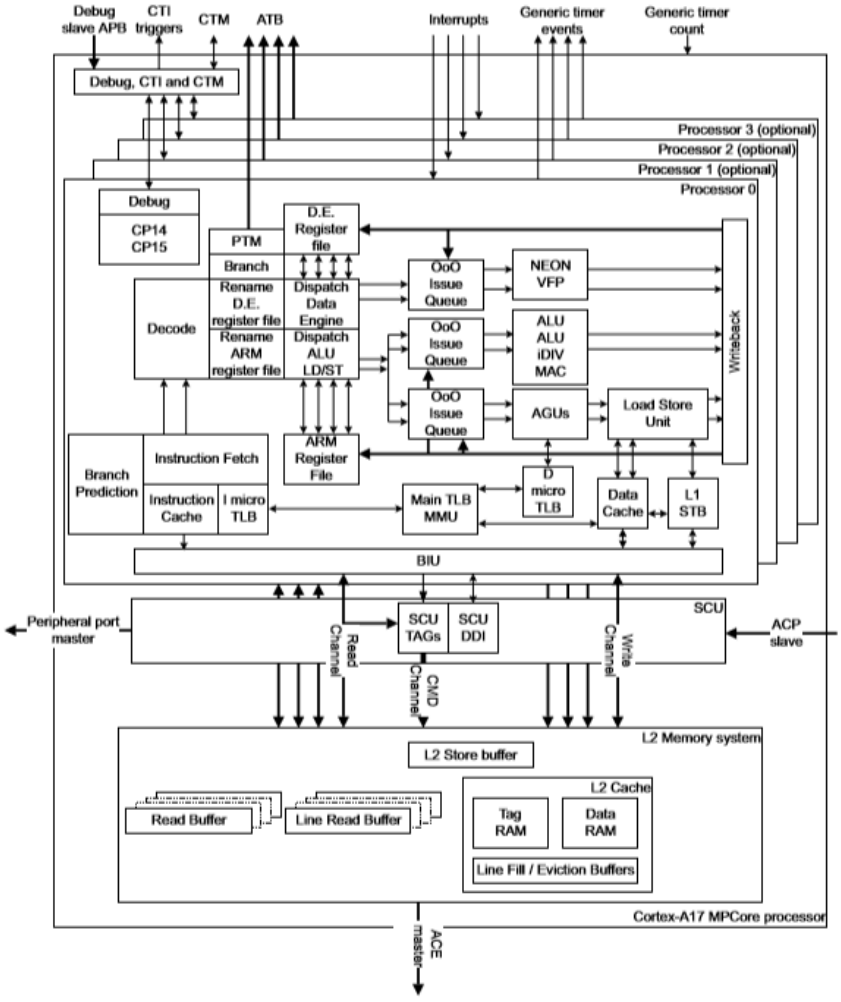
VREV16 (Vector Reverse in halfwords) reverses the order of 8-bit elements in each halfword of the vector, and places the result in the corresponding destination vector.

VREV32 (Vector Reverse in words) reverses the order of 8-bit or 16-bit elements in each word of the vector, and places the result in the corresponding destination vector.

VREV64 (Vector Reverse in doublewords) reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector, and places the result in the corresponding destination vector.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf



Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

The Cortex-A17 MPCore processor implements the ARMv7-A profile architecture with the following architecture extensions:

- *Advanced Single Instruction Multiple Data version 2* (SIMDv2) architecture extension for integer and floating-point vector operations.

  Note

  The Advanced SIMD architecture extension, its associated implementations, and supporting software, are commonly referred to as NEON technology.

- *Vector Floating-Point version 4* (VFPv4) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard.
- Security Extensions for implementation of enhanced security.
- Virtualization Extensions for the development of virtualized systems that enable the switching of guest operating systems.
- *Large Physical Address Extension* (LPAE) for address translation of up to 40-bit physical addresses.
- Multiprocessing Extensions for multiprocessing functionality.

Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

Table A4-24 Miscellaneous Advanced SIMD data-processing instructions

Instruction	See
Vector Absolute Difference and Accumulate	*VABA, VABAL* on page A8-819
Vector Absolute Difference	*VABD, VABDL (integer)* on page A8-821 *VABD (floating-point)* on page A8-823
Vector Absolute	*VABS* on page A8-825
Vector Convert between floating-point and fixed point	*VCVT (between floating-point and fixed-point, Advanced SIMD)* on page A8-873
Vector Convert between floating-point and integer	*VCVT (between floating-point and integer, Advanced SIMD)* on page A8-869
Vector Convert between half-precision and single-precision	*VCVT (between half-precision and single-precision, Advanced SIMD)* on page A8-879
Vector Count Leading Sign Bits	*VCLS* on page A8-859
Vector Count Leading Zeros	*VCLZ* on page A8-863
Vector Count Set Bits	*VCNT* on page A8-867
Vector Duplicate scalar	*VDUP (scalar)* on page A8-885
Vector Extract	*VEXT* on page A8-891
Vector Move and Narrow	*VMOVN* on page A8-953
Vector Move Long	*VMOVL* on page A8-951
Vector Maximum, Minimum	*VMAX, VMIN (integer)* on page A8-927 *VMAX, VMIN (floating-point)* on page A8-929
Vector Negate	*VNEG* on page A8-969
Vector Pairwise Maximum, Minimum	*VPMAX, VPMIN (integer)* on page A8-987 *VPMAX, VPMIN (floating-point)* on page A8-989
Vector Reciprocal Estimate	*VRECPE* on page A8-1025
Vector Reciprocal Step	*VRECPS* on page A8-1027
Vector Reciprocal Square Root Estimate	*VRSQRTE* on page A8-1039
Vector Reciprocal Square Root Step	*VRSQRTS* on page A8-1041
Vector Reverse	*VREV16, VREV32, VREV64* on page A8-1029
Vector Saturating Absolute	*VQABS* on page A8-995
Vector Saturating Move and Narrow	*VQMOVN, VQMOVUN* on page A8-1005
Vector Saturating Negate	*VQNEG* on page A8-1007
Vector Swap	*VSWP* on page A8-1093
Vector Table Lookup	*VTBL, VTBX* on page A8-1095

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

VREV16, VREV32, VREV64

VREV16 (Vector Reverse in halfwords) reverses the order of 8-bit elements in each halfword of the vector, and places the result in the corresponding destination vector.

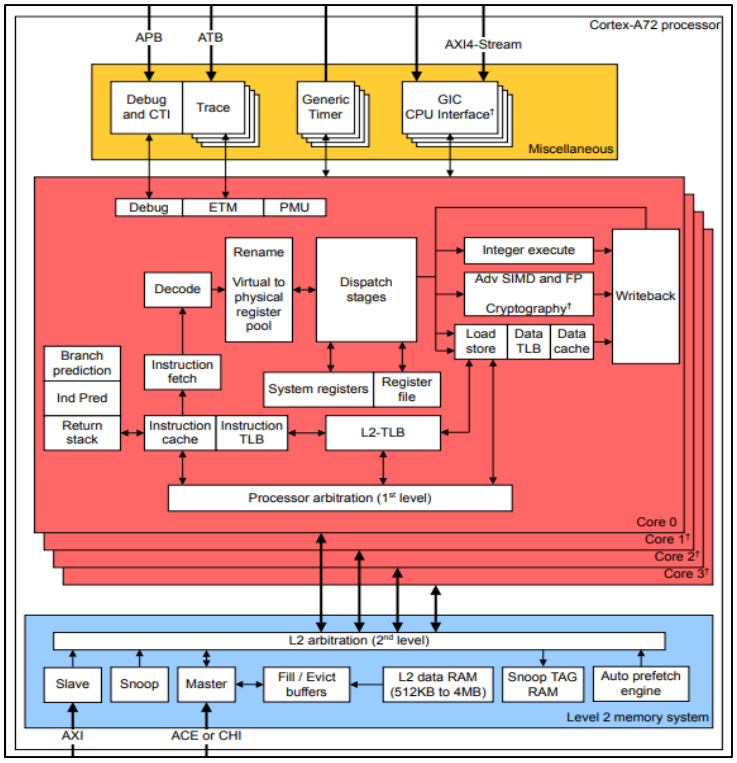
VREV32 (Vector Reverse in words) reverses the order of 8-bit or 16-bit elements in each word of the vector, and places the result in the corresponding destination vector.

VREV64 (Vector Reverse in doublewords) reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector, and places the result in the corresponding destination vector.

There is no distinction between data types, other than size.

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf



Source: ARM Cortex-A72 MPCore Processor manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.100095_0003_06_en/cortex_a72_mpcore_trm_100095_0003_06_en.pdf

Advanced SIMD and Floating-point unit

The Advanced SIMD and Floating-point unit provides support for the ARMv8 Advanced SIMD and Floating-point execution. In addition, the Advanced SIMD and Floating-point unit provides support for the optional Cryptography engine.

Source: ARM Cortex-A72 MPCore Processor manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.100095_0003_06_en/cortex_a72_mpcore_trm_100095_0003_06_en.pdf

Table B2-2 Byte reversal instructions

Function	Instructions	Notes
Reverse bytes in 32-bit word or words[a]	REV32	For use with general-purpose registers
Reverse bytes in whole register	REV	For use with general-purpose registers
Reverse bytes in 16-bit halfwords	REV16	For use with general-purpose registers
Reverse elements in doublewords, vector	REV64	For use with SIMD and floating-point registers
Reverse elements in words, vector	REV32	For use with SIMD and floating-point registers
Reverse elements in halfwords, vector	REV16	For use with SIMD and floating-point registers

a. Can operate on multiple words.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

REV64

Reverse elements in 64-bit doublewords (vector). This instruction reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector in the source SIMD&FP register, places the results into a vector, and writes the vector to the destination SIMD&FP register.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

NEON

Arm NEON technology is an advanced SIMD (single instruction multiple data) architecture extension for the Arm Cortex-A series and Cortex-R52 processors.

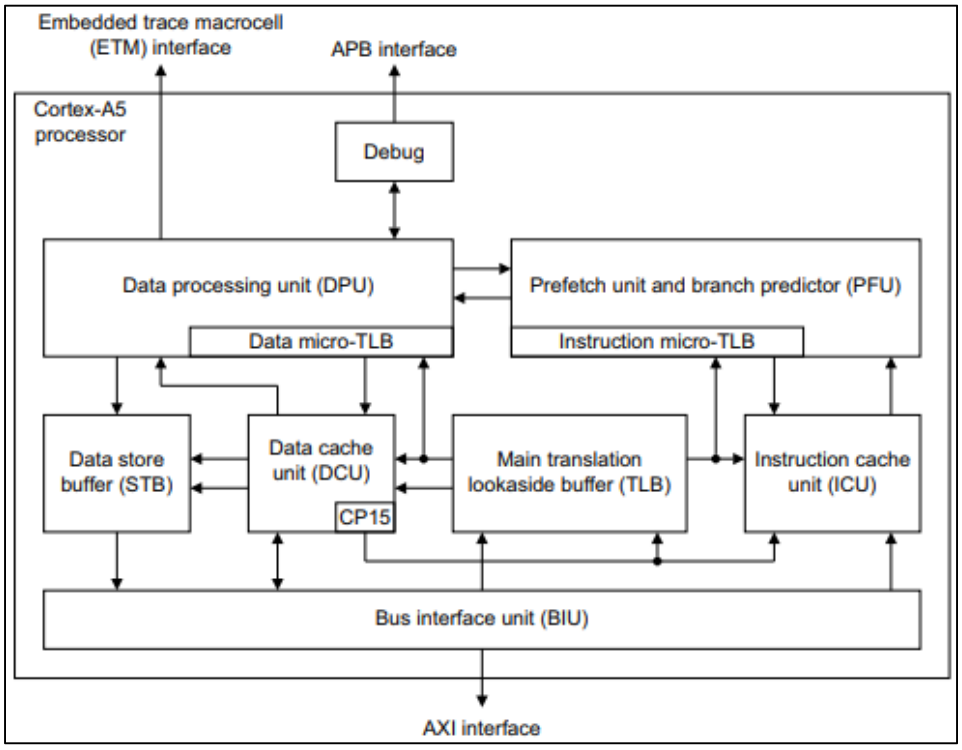
Source: https://developer.arm.com/technologies/neon

Examples of some key available functions are detailed below:

Video codecs:	Audio codecs:	Voice and speech codecs:	Audio enhancement algorithms:	Computer Vision	Machine and deep learning
VP9 OTT encoder, VP9 Consumer encoder/decoder	MP3 encoder/decoder	G.711	Echo cancellation	Canny Edge detection	On-device object recognition
H.264 (AVC) encoder/decoder	MPEG-2 layer I & II encoder/decoder	G.722, G.722.1, G.722.2	Noise Reduction	Harris Corner	On-device scene recognition
MPEG4 SP/ASP encoder/decoder	MPEG-1 layer III audio encoder	G.723.1	Beam Forming	ORB	Human pose recognition
MPEG2 decoder	MPEG-1 layer III audio encoder/decoder	G.726	Comfort Noise	Convolution filter	Defect detection
H.263 decoder	HE-AACv1, v2 encoder/decoder	G.727	AudioZoom	Erosion/Dilation	

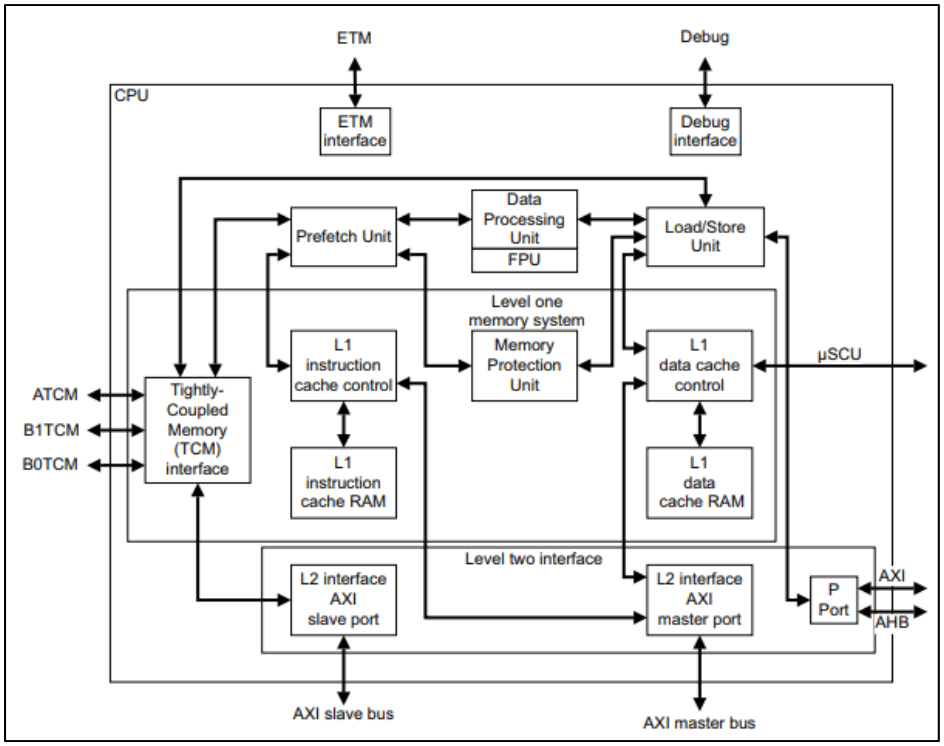
Source: https://developer.arm.com/technologies/neon

36. The accused products include a central processing unit coupled to the instruction buffer, the memory, and the getbits processing engine.



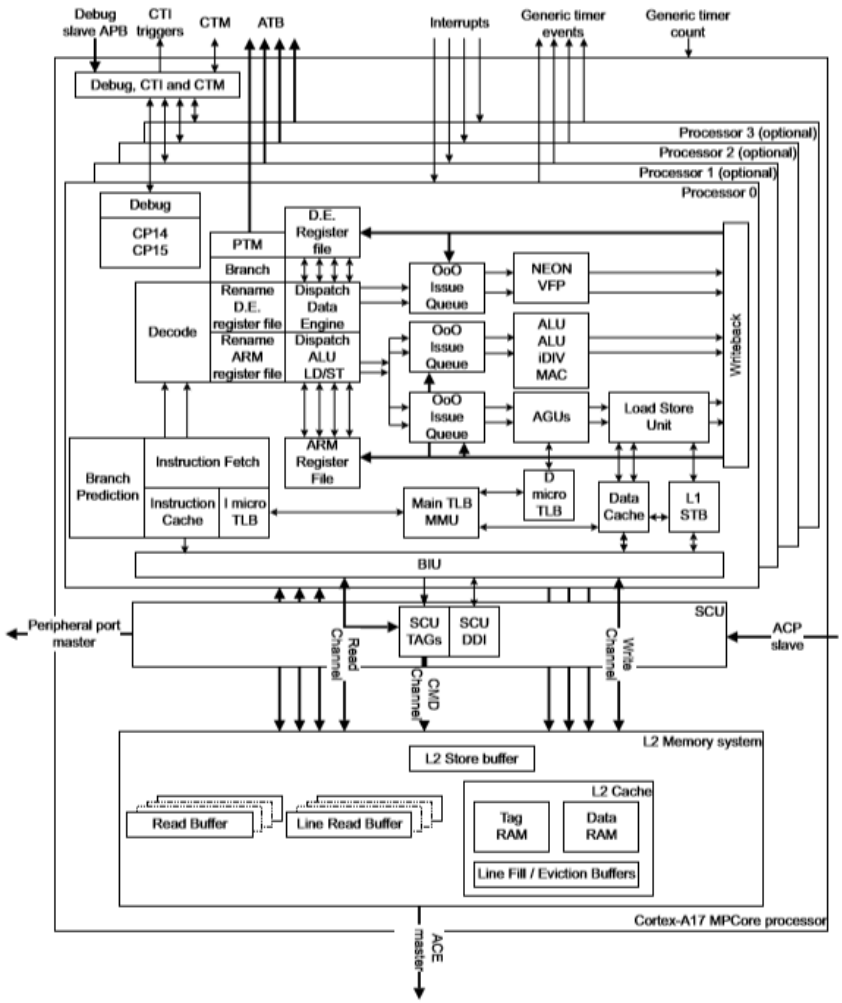
Source: ARM Cortex-A5 Technical Reference Manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0433c/DDI0433C_cortex_a5_trm.pdf



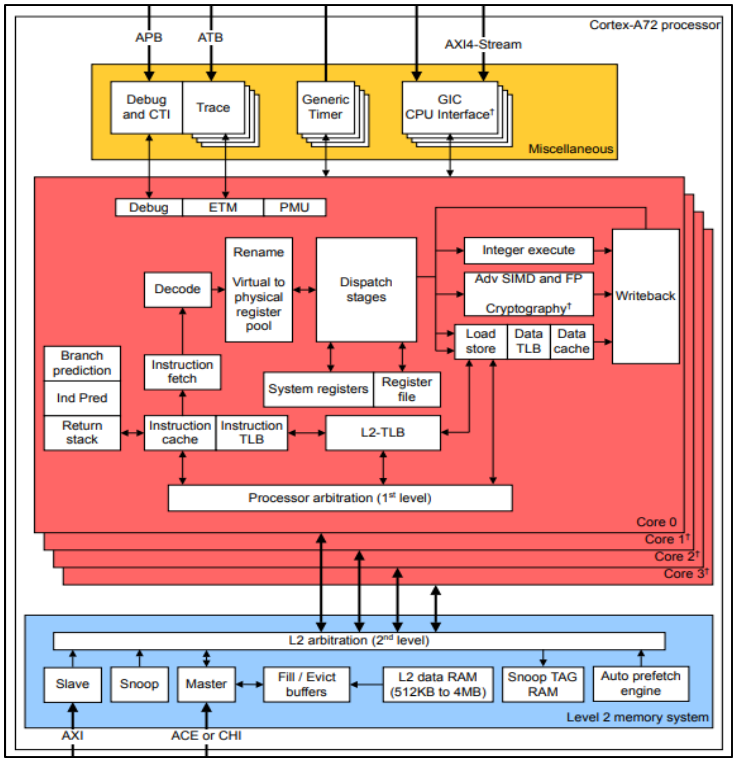
Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf



Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf



Source: ARM Cortex-A72 MPCore Processor manual downloaded from http://infocenter.arm.com/help/topic/com.arm.doc.100095_0003_06_en/cortex_a72_mpcore_trm_100095_0003_06_en.pdf

37. Defendants have had knowledge of the '293 Patent at least as of the date when they were notified of the filing of this action.

38. American Patents has been damaged as a result of the infringing conduct by Defendants alleged above. Thus, Defendants are liable to American Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

39. American Patents and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the '293 Patent.

COUNT II

DIRECT INFRINGEMENT OF U.S. PATENT NO. 6,587,058

40. On July 1, 2003, United States Patent No. 6,587,058 ("the '058 Patent") was duly and legally issued by the United States Patent and Trademark Office for an invention entitled "Processing Circuit And Method For Variable-Length Coding And Decoding."

41. American Patents is the owner of the '058 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the '058 Patent against infringers, and to collect damages for all relevant times.

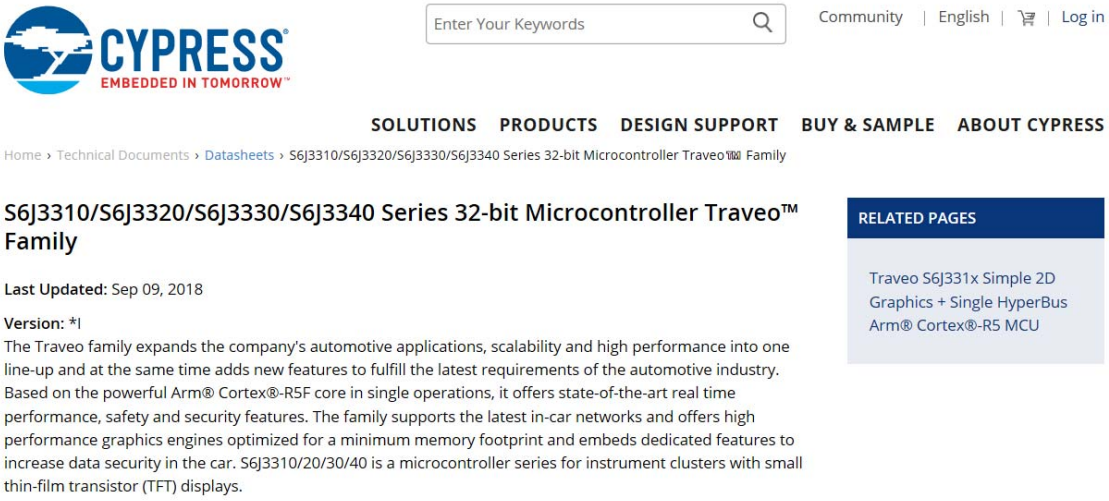
42. Analog made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its ADSP-SC584 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[5]:

[5] A non-exhaustive list of additional accused products includes the ADSP-SC587 and ADSP-SC589 families of products that include advanced circuits for variable-length coding and decoding of media data.

Source: https://www.analog.com/en/products/adsp-sc584.html#product-overview

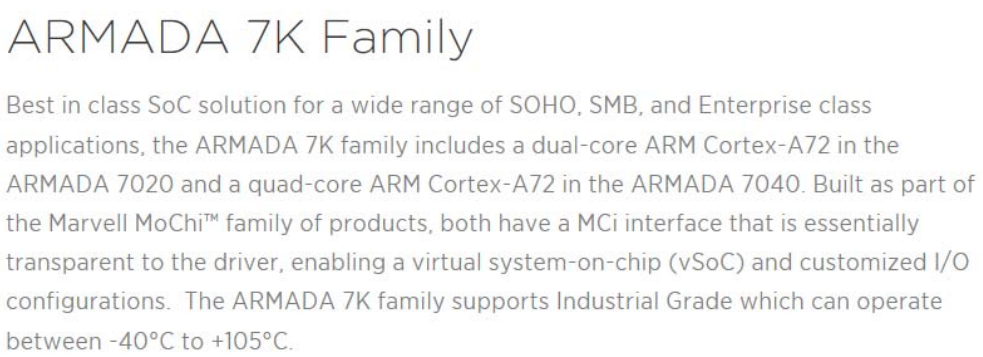
43. Cypress made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Traveo S6J3310 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[6]:

[6] A non-exhaustive list of additional accused products includes the Traveo S6J3320 and S6J3340 families of products that include advanced circuits for variable-length coding and decoding of media data.



Source: http://www.cypress.com/documentation/datasheets/s6j3310s6j3320s6j3330s6j3340-series-32-bit-microcontroller-traveo-family

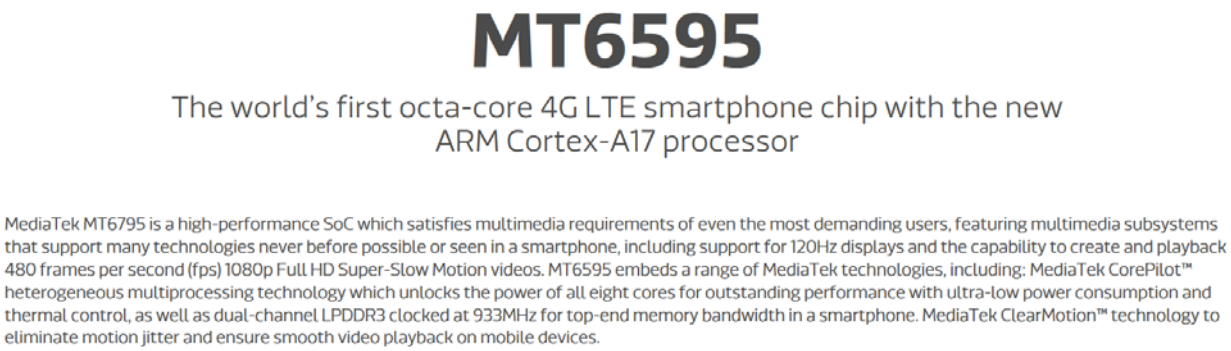
44. Marvell made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its ARMADA 7020 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[7]:



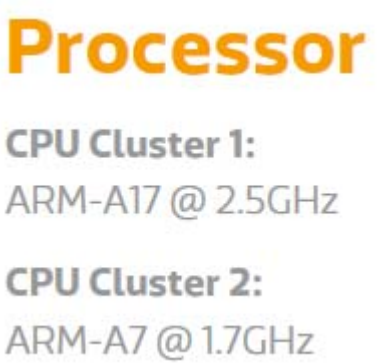
Source: https://www.marvell.com/embedded-processors/armada-70xx/

[7] A non-exhaustive list of additional accused products includes the ARMADA 7040, ARMADA 8020, and ARMADA 8040 families of products that include advanced circuits for variable-length coding and decoding of media data.

45. MediaTek made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its MT6595 and Helio X27 families of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[8]:



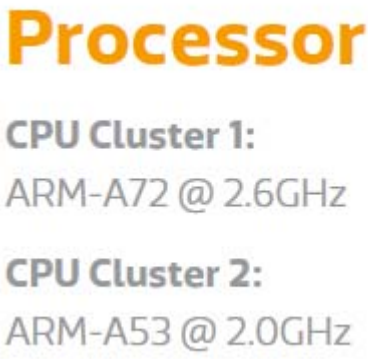
Source: https://www.mediatek.com/products/smartphones/mt6595



Source: https://www.mediatek.com/products/smartphones/mt6595

[8] A non-exhaustive list of additional accused products includes the Helio P60, Helio P70, MT8173, MT8176, and Helio X series (including Helio X20, Helio X23, and Helio X25) families of products that include advanced circuits for variable-length coding and decoding of media data.

Source: https://www.mediatek.com/products/smartphones/mt6797x-helio-x27



Source: https://www.mediatek.com/products/smartphones/mt6797x-helio-x27

46. By doing so, Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least Claim 1 of the '058 Patent. Defendants' infringement in this regard is ongoing.

47. Defendants have infringed the '058 Patent by making, having made, using, importing, providing, supplying, distributing, selling or offering for sale integrated circuits having variable-length encode/decode processors.

48. The accused products include a central processing unit and an instruction buffer coupled to the central processing unit.

49. The accused products include a getbits processing engine coupled to the central processing unit.

Media Processing Engine

The MPE implements ARM NEON technology, a media and signal processing architecture that adds instructions targeted at audio, video, 3-D graphics, image, and speech processing. Advanced SIMD instructions are available in both ARM and Thumb states.

If the design includes the MPE, the FPU is included.

See the *Cortex-A5 NEON Media Processing Engine Technical Reference Manual*.

Source: ARM Cortex-A5 Technical Reference Manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0433c/DDI0433C_cortex_a5_trm.pdf

The Cortex-A5 processor implements the ARM v7-A architecture profile. This includes:

- the 32-bit ARM instruction set
- the Thumb instruction set, a variable-length instruction set, that has both 16-bit and 32-bit instructions
- execution of 8-bit Java bytecodes
- the ThumbEE instruction set
- the security extensions, TrustZone
- the VFPv4 Floating point architecture and Advanced SIMD extensions, NEON.

Source: ARM Cortex-A5 Technical Reference Manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0433c/DDI0433C_cortex_a5_trm.pdf

ARM architecture

The Cortex-R5 processor implements the ARMv7-R architecture profile that includes the following architecture extensions:

- Advanced *Single Instruction Multiple Data* (SIMD) architecture extension for integer and floating-point vector operations
- *Vector Floating-Point version 3* (VFPv3) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard
- Multiprocessing Extensions for multiprocessing functionality.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

Note

- The Cortex-R5F processor is a Cortex-R5 processor that includes the optional *Floating Point Unit* (FPU) extension.
- In this book, references to the Cortex-R5 processor also apply to the Cortex-R5F processor, unless the context makes it clear that this is not the case.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

Table A4-24 Miscellaneous Advanced SIMD data-processing instructions

Instruction	See
Vector Absolute Difference and Accumulate	*VABA, VABAL* on page A8-819
Vector Absolute Difference	*VABD, VABDL (integer)* on page A8-821 *VABD (floating-point)* on page A8-823
Vector Absolute	*VABS* on page A8-825
Vector Convert between floating-point and fixed point	*VCVT (between floating-point and fixed-point, Advanced SIMD)* on page A8-873
Vector Convert between floating-point and integer	*VCVT (between floating-point and integer, Advanced SIMD)* on page A8-869
Vector Convert between half-precision and single-precision	*VCVT (between half-precision and single-precision, Advanced SIMD)* on page A8-879
Vector Count Leading Sign Bits	*VCLS* on page A8-859
Vector Count Leading Zeros	*VCLZ* on page A8-863
Vector Count Set Bits	*VCNT* on page A8-867
Vector Duplicate scalar	*VDUP (scalar)* on page A8-885
Vector Extract	*VEXT* on page A8-891
Vector Move and Narrow	*VMOVN* on page A8-953
Vector Move Long	*VMOVL* on page A8-951
Vector Maximum, Minimum	*VMAX, VMIN (integer)* on page A8-927 *VMAX, VMIN (floating-point)* on page A8-929
Vector Negate	*VNEG* on page A8-969
Vector Pairwise Maximum, Minimum	*VPMAX, VPMIN (integer)* on page A8-987 *VPMAX, VPMIN (floating-point)* on page A8-989
Vector Reciprocal Estimate	*VRECPE* on page A8-1025
Vector Reciprocal Step	*VRECPS* on page A8-1027
Vector Reciprocal Square Root Estimate	*VRSQRTE* on page A8-1039
Vector Reciprocal Square Root Step	*VRSQRTS* on page A8-1041
Vector Reverse	*VREV16, VREV32, VREV64* on page A8-1029
Vector Saturating Absolute	*VQABS* on page A8-995
Vector Saturating Move and Narrow	*VQMOVN, VQMOVUN* on page A8-1005
Vector Saturating Negate	*VQNEG* on page A8-1007
Vector Swap	*VSWP* on page A8-1093
Vector Table Lookup	*VTBL, VTBX* on page A8-1095

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

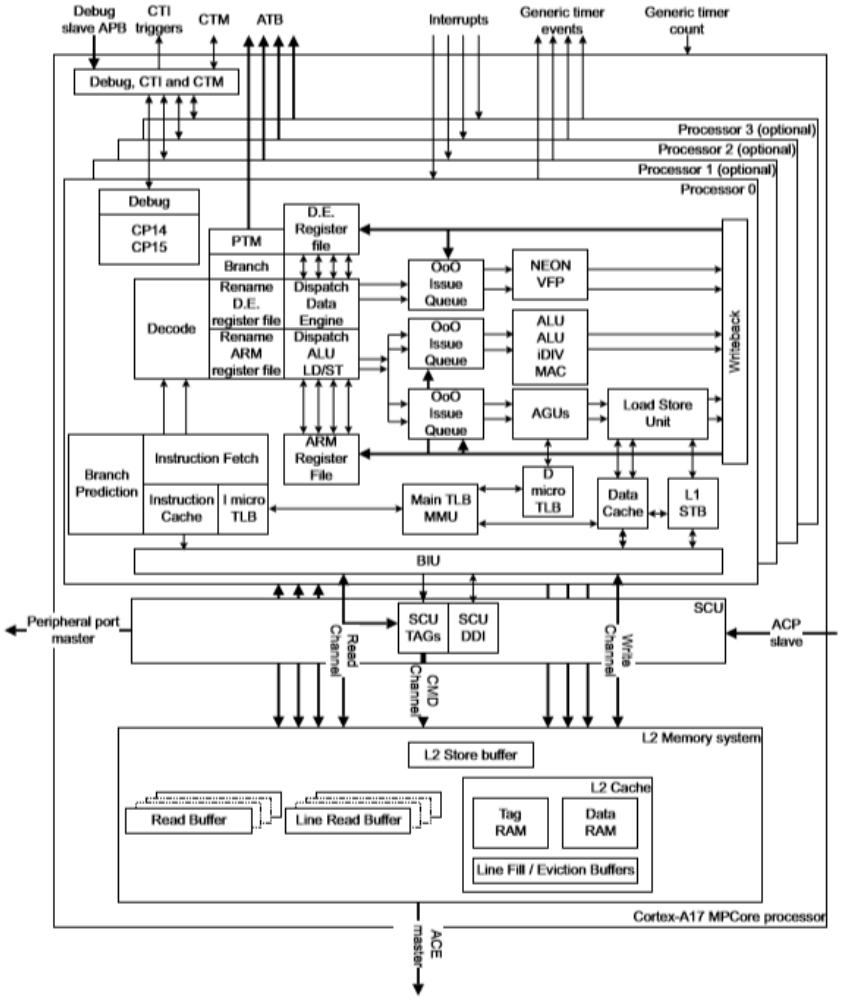
VREV16 (Vector Reverse in halfwords) reverses the order of 8-bit elements in each halfword of the vector, and places the result in the corresponding destination vector.

VREV32 (Vector Reverse in words) reverses the order of 8-bit or 16-bit elements in each word of the vector, and places the result in the corresponding destination vector.

VREV64 (Vector Reverse in doublewords) reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector, and places the result in the corresponding destination vector.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf



Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

The Cortex-A17 MPCore processor implements the ARMv7-A profile architecture with the following architecture extensions:

- *Advanced Single Instruction Multiple Data version 2* (SIMDv2) architecture extension for integer and floating-point vector operations.

  Note

  The Advanced SIMD architecture extension, its associated implementations, and supporting software, are commonly referred to as NEON technology.

- *Vector Floating-Point version 4* (VFPv4) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard.
- Security Extensions for implementation of enhanced security.
- Virtualization Extensions for the development of virtualized systems that enable the switching of guest operating systems.
- *Large Physical Address Extension* (LPAE) for address translation of up to 40-bit physical addresses.
- Multiprocessing Extensions for multiprocessing functionality.

Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

Table A4-24 Miscellaneous Advanced SIMD data-processing instructions

Instruction	See
Vector Absolute Difference and Accumulate	*VABA, VABAL* on page A8-819
Vector Absolute Difference	*VABD, VABDL (integer)* on page A8-821 *VABD (floating-point)* on page A8-823
Vector Absolute	*VABS* on page A8-825
Vector Convert between floating-point and fixed point	*VCVT (between floating-point and fixed-point, Advanced SIMD)* on page A8-873
Vector Convert between floating-point and integer	*VCVT (between floating-point and integer, Advanced SIMD)* on page A8-869
Vector Convert between half-precision and single-precision	*VCVT (between half-precision and single-precision, Advanced SIMD)* on page A8-879
Vector Count Leading Sign Bits	*VCLS* on page A8-859
Vector Count Leading Zeros	*VCLZ* on page A8-863
Vector Count Set Bits	*VCNT* on page A8-867
Vector Duplicate scalar	*VDUP (scalar)* on page A8-885
Vector Extract	*VEXT* on page A8-891
Vector Move and Narrow	*VMOVN* on page A8-953
Vector Move Long	*VMOVL* on page A8-951
Vector Maximum, Minimum	*VMAX, VMIN (integer)* on page A8-927 *VMAX, VMIN (floating-point)* on page A8-929
Vector Negate	*VNEG* on page A8-969
Vector Pairwise Maximum, Minimum	*VPMAX, VPMIN (integer)* on page A8-987 *VPMAX, VPMIN (floating-point)* on page A8-989
Vector Reciprocal Estimate	*VRECPE* on page A8-1025
Vector Reciprocal Step	*VRECPS* on page A8-1027
Vector Reciprocal Square Root Estimate	*VRSQRTE* on page A8-1039
Vector Reciprocal Square Root Step	*VRSQRTS* on page A8-1041
Vector Reverse	*VREV16, VREV32, VREV64* on page A8-1029
Vector Saturating Absolute	*VQABS* on page A8-995
Vector Saturating Move and Narrow	*VQMOVN, VQMOVUN* on page A8-1005
Vector Saturating Negate	*VQNEG* on page A8-1007
Vector Swap	*VSWP* on page A8-1093
Vector Table Lookup	*VTBL, VTBX* on page A8-1095

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

VREV16, VREV32, VREV64

VREV16 (Vector Reverse in halfwords) reverses the order of 8-bit elements in each halfword of the vector, and places the result in the corresponding destination vector.

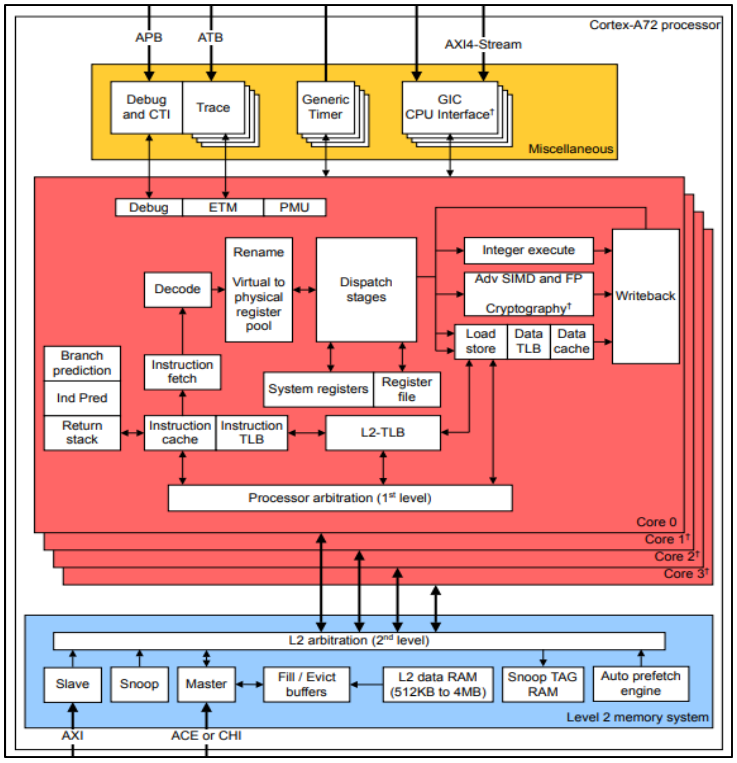
VREV32 (Vector Reverse in words) reverses the order of 8-bit or 16-bit elements in each word of the vector, and places the result in the corresponding destination vector.

VREV64 (Vector Reverse in doublewords) reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector, and places the result in the corresponding destination vector.

There is no distinction between data types, other than size.

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf



Source: ARM Cortex-A72 MPCore Processor manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.100095_0003_06_en/cortex_a72_mpcore_trm_100095_0003_06_en.pdf

Advanced SIMD and Floating-point unit

The Advanced SIMD and Floating-point unit provides support for the ARMv8 Advanced SIMD and Floating-point execution. In addition, the Advanced SIMD and Floating-point unit provides support for the optional Cryptography engine.

Source: ARM Cortex-A72 MPCore Processor manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.100095_0003_06_en/cortex_a72_mpcore_trm_100095_0003_06_en.pdf

Table B2-2 Byte reversal instructions

Function	Instructions	Notes
Reverse bytes in 32-bit word or words[a]	REV32	For use with general-purpose registers
Reverse bytes in whole register	REV	For use with general-purpose registers
Reverse bytes in 16-bit halfwords	REV16	For use with general-purpose registers
Reverse elements in doublewords, vector	REV64	For use with SIMD and floating-point registers
Reverse elements in words, vector	REV32	For use with SIMD and floating-point registers
Reverse elements in halfwords, vector	REV16	For use with SIMD and floating-point registers

a. Can operate on multiple words.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

REV64

Reverse elements in 64-bit doublewords (vector). This instruction reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector in the source SIMD&FP register, places the results into a vector, and writes the vector to the destination SIMD&FP register.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

NEON

Arm NEON technology is an advanced SIMD (single instruction multiple data) architecture extension for the Arm Cortex-A series and Cortex-R52 processors.

Source: https://developer.arm.com/technologies/neon

Examples of some key available functions are detailed below:

Video codecs:	Audio codecs:	Voice and speech codecs:	Audio enhancement algorithms:	Computer Vision	Machine and deep learning
VP9 OTT encoder, VP9 Consumer encoder/decoder	MP3 encoder/decoder	G.711	Echo cancellation	Canny Edge detection	On-device object recognition
H.264 (AVC) encoder/decoder	MPEG-2 layer I & II encoder/decoder	G.722, G.722.1, G.722.2	Noise Reduction	Harris Corner	On-device scene recognition
MPEG4 SP/ASP encoder/decoder	MPEG-1 layer III audio encoder	G.723.1	Beam Forming	ORB	Human pose recognition
MPEG2 decoder	MPEG-1 layer III audio encoder/decoder	G.726	Comfort Noise	Convolution filter	Defect detection
H.263 decoder	HE-AACv1, v2 encoder/decoder	G.727	AudioZoom	Erosion/Dilation	

Source: https://developer.arm.com/technologies/neon

50. The accused products include at least one shared register coupled to the central processing unit and to the getbits processing engine.

About the Cortex-A5 NEON MPE

The Cortex-A5 NEON MPE extends the Cortex-A5 functionality to provide support for the ARM v7 Advanced SIMD v2 and *Vector Floating-Point v4* (VFPv4) instruction sets.

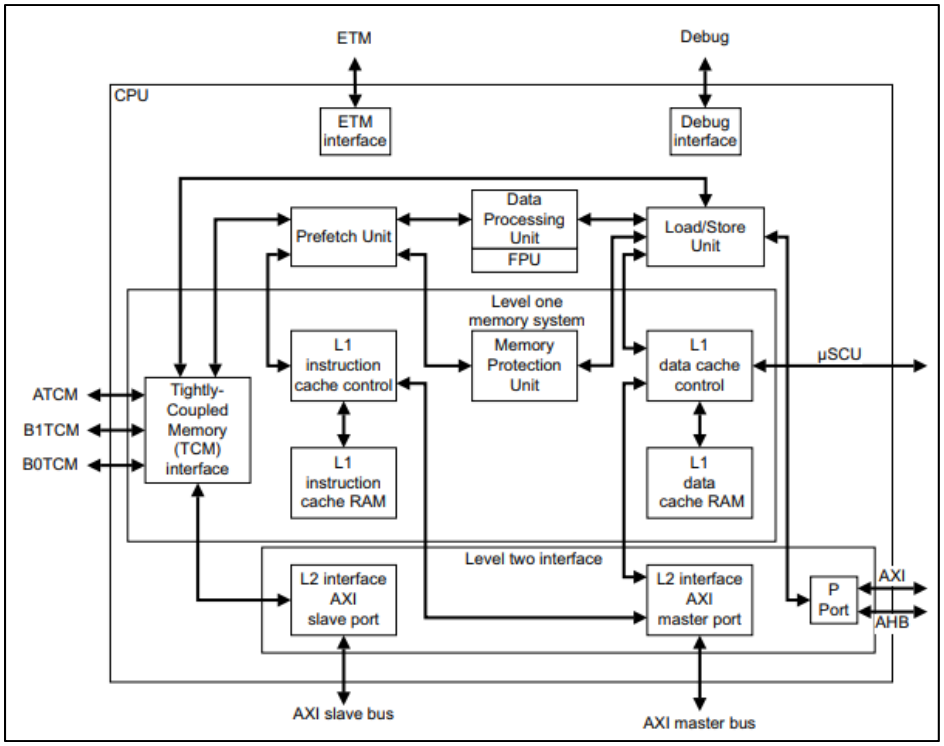
The NEON MPE supports all addressing modes and operations that are described in the *ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

The NEON MPE features are:

- SIMD and scalar single-precision floating-point computation.
- Scalar double-precision floating-point computation.
- SIMD and scalar half-precision floating-point conversion.
- SIMD 8, 16, 32, and 64-bit signed and unsigned integer computation.
- 8 or 16-bit polynomial computation for single-bit coefficients.
- Structured data load capabilities.
- Large, shared register file, addressable as:
  - Thirty-two 32-bit S (single) registers.
  - Thirty-two 64-bit D (double) registers.
  - Sixteen 128-bit Q (quad) registers See the *ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition* for details of the extension register set.

Source: ARM Cortex-A5 NEON Media Processing Engine Technical Reference Manual downloaded from

https://static.docs.arm.com/100304/0001/cortex_a5_neon_mpe_trm_100304_0001_00_en.pdf



Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

The DPU holds most of the program-visible state of the processor, such as general-purpose registers, status registers and control registers. It decodes and executes instructions, operating on data held in the registers in accordance with the ARM architecture. Instructions are fed to the DPU from the PFU through a buffer. The DPU performs instructions that require data to be transferred to or from the memory system by interfacing to the LSU. See Chapter 3 *Programmers Model* for more information.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

Advanced SIMD and Floating-point system registers

The Advanced SIMD and Floating-point (VFP) Extensions have a shared register space for system registers. Only one register in this space is accessible at the Application level, see either:

- *FPSCR, Floating-point Status and Control Register, VMSA* on page B4-1566.
- *FPSCR, Floating-point Status and Control Register, PMSA* on page B6-1839.

Note

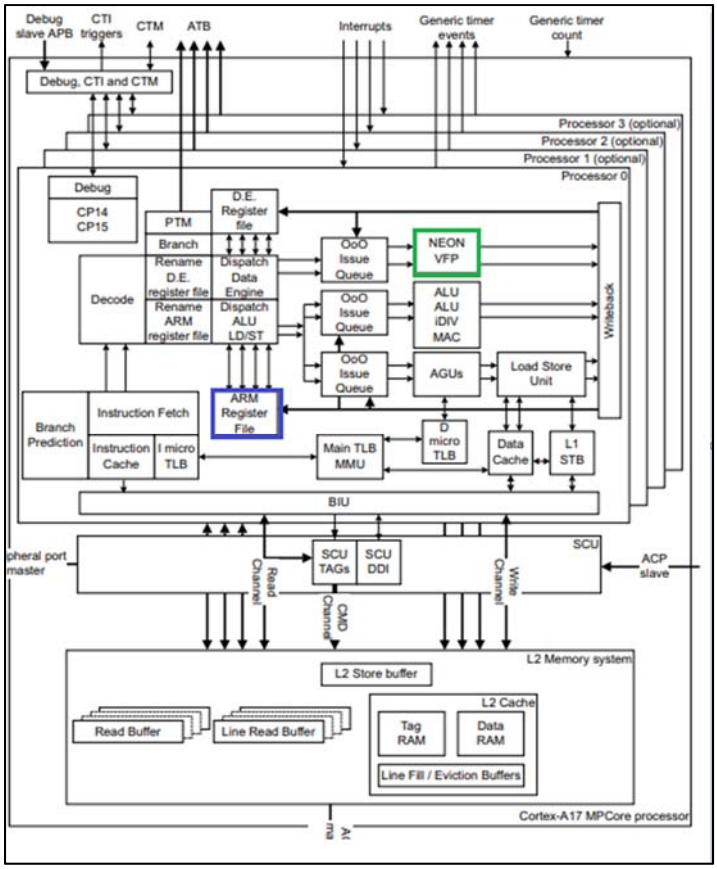
In this chapter, short links to the FPSCR are to the description in Chapter B4 *System Control Registers in a VMSA implementation*. The FPSCR description in Chapter B6 *System Control Registers in a PMSA implementation* is identical to this description.

Writes to the FPSCR can have side-effects on various aspects of processor operation. All of these side-effects are synchronous to the FPSCR write. This means they are guaranteed not to be visible to earlier instructions in the execution stream, and they are guaranteed to be visible to later instructions in the execution stream.

See *Advanced SIMD and Floating-point Extension system registers* on page B1-1235 for the system level view of the registers.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf



Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

Instruction sets, arithmetic operations, and register files

The ARM and Thumb instruction sets both provide a wide range of integer arithmetic and logical operations, that operate on register file of sixteen 32-bit registers, the *ARM core registers*. As described in *ARM core registers* on page A2-45, these registers include the special registers SP, LR, and PC. *ARM core data types and arithmetic* on page A2-40 gives more information about these operations.

In addition, if an implementation includes:

- The Floating-point (VFP) Extension, the ARM and Thumb instruction sets include floating-point instructions.
- The Advanced SIMD Extension, the ARM and Thumb instruction sets include vector instructions.

Floating-point and vector instructions operate on an independent register file, described in *Advanced SIMD and Floating-point Extension registers* on page A2-56. In an implementation that includes both of these extensions, they share a common register file. The following sections give more information about these extensions and the instructions they provide:

- *Advanced SIMD and Floating-point Extensions* on page A2-54.
- *Floating-point data types and arithmetic* on page A2-62.
- *Polynomial arithmetic over {0, 1}* on page A2-92.

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

The processor supports all addressing modes, data types, and operations in the VFPv4 extension with version 3 of the Common VFP subarchitecture. The processor implements VFPv4-D32. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on the VFPv4 instruction set.

Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

From VFPv3, the Advanced SIMD and Floating-point (VFP) Extensions use the same register set. This is distinct from the ARM core register set. These registers are generally referred to as the *extension registers*.

The extension register set consists of either 32 or 16 doubleword registers, as follows:

- If VFPv2 is implemented, it consists of 16 doubleword registers.
- If VFPv3 is implemented, it consists of either 32 or 16 doubleword registers. Where necessary, these two implementation options are distinguished using the terms:
  - VFPv3-D32, for an implementation with 32 doubleword registers.
  - VFPv3-D16, for an implementation with 16 doubleword registers.
- If VFPv4 is implemented, it consists of either 32 or 16 doubleword registers. Where necessary, these two implementation options are distinguished using the terms:
  - VFPv4-D32, for an implementation with 32 doubleword registers.
  - VFPv4-D16, for an implementation with 16 doubleword registers.

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

The ARMv8 architecture provides two register files:

- A general-purpose register file.
- A SIMD&FP register file.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

Vector formats

In an implementation that includes the SIMD instructions that operate on the SIMD&FP register file, a register can hold one or more packed elements, all of the same size and type. The combination of a register and a data type describes a vector of elements. The vector is considered to be an array of elements of the data type specified in the instruction. The number of elements in the vector is implied by the size of the data elements and the size of the register.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

TBL

Table vector Lookup. This instruction reads each value from the vector elements in the index source SIMD&FP register, uses each result as an index to perform a lookup in a table of bytes that is described by one to four source table SIMD&FP registers, places the lookup result in a vector, and writes the vector to the destination SIMD&FP register. If an index is out of range for the table, the result for that lookup is 0. If more than one source register is used to describe the table, the first source register describes the lowest bytes of the table.

Source: ARMv8-A Architecture Reference Manual downloaded from https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

SIMD vector register list

Where an instruction operates on multiple SIMD and floating-point registers, for example vector Load/Store structure and table lookup operations, the registers are specified as a list enclosed by curly braces. This list consists of either a sequence of registers separated by commas, or a register range separated by a hyphen. The registers must be numbered in increasing order, modulo 32, in increments of one. The hyphenated form is preferred for disassembly if there are more than two registers in the list and the register number are increasing. The following examples are equivalent representations of a set of four registers V4 to V7, each holding four lanes of 32-bit elements:

Source: ARMv8-A Architecture Reference Manual downloaded from https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

51. Defendants have had knowledge of the ‘058 Patent at least as of the date when they were notified of the filing of this action.

52. American Patents has been damaged as a result of the infringing conduct by Defendants alleged above. Thus, Defendants are liable to American Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

53. American Patents and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the ‘058 Patent.

COUNT III

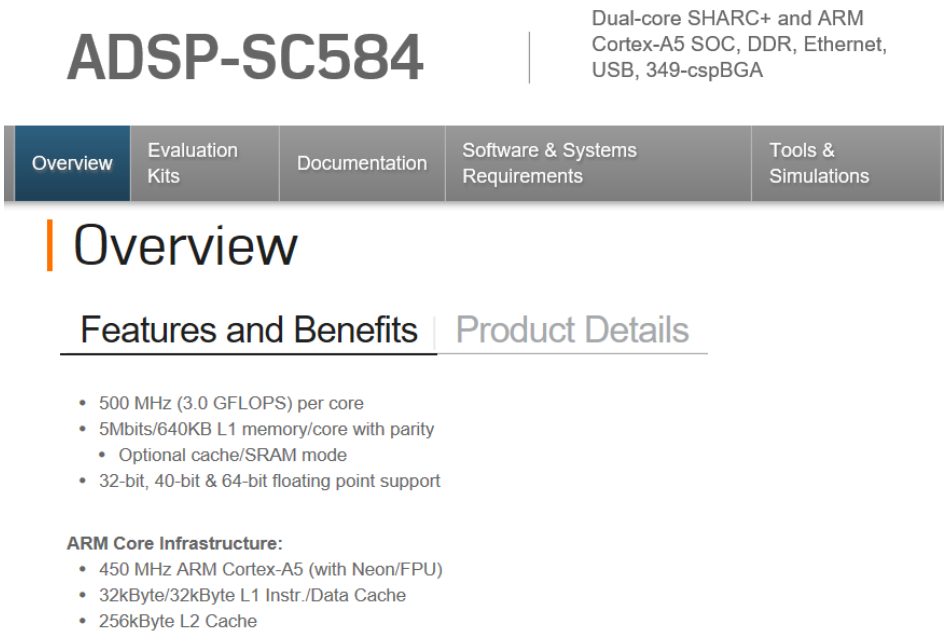
DIRECT INFRINGEMENT OF U.S. PATENT NO. 7,262,720

54. On August 28, 2007, United States Patent No. 7,262,720 ("the '720 Patent") was duly and legally issued by the United States Patent and Trademark Office for an invention entitled "Processing Circuit And Method For Variable-Length Coding And Decoding."

55. American Patents is the owner of the '720 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the '720 Patent against infringers, and to collect damages for all relevant times.

56. Analog made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its ADSP-SC584 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products")[9]:

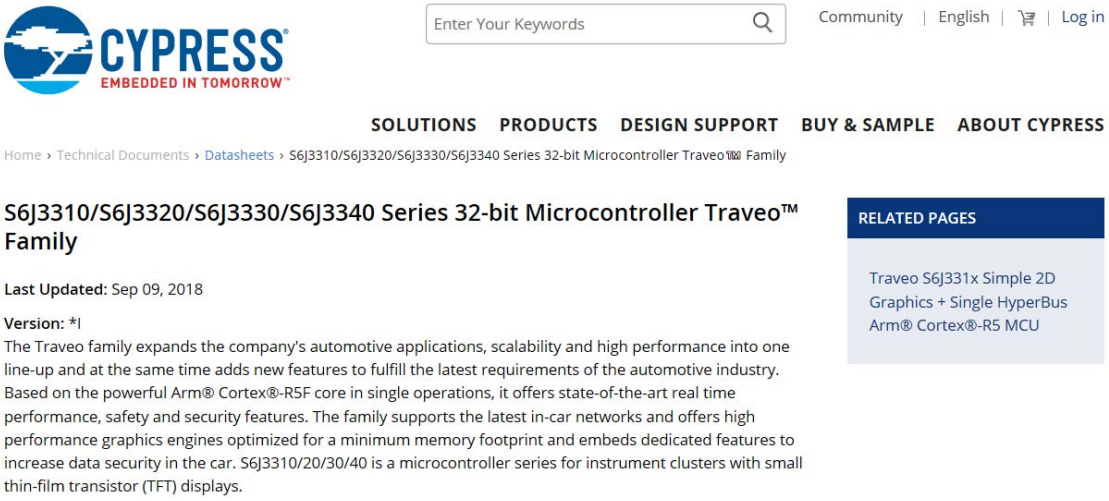
[9] A non-exhaustive list of additional accused products includes the ADSP-SC587 and ADSP-SC589 families of products that include advanced circuits for variable-length coding and decoding of media data.



Source: https://www.analog.com/en/products/adsp-sc584.html#product-overview

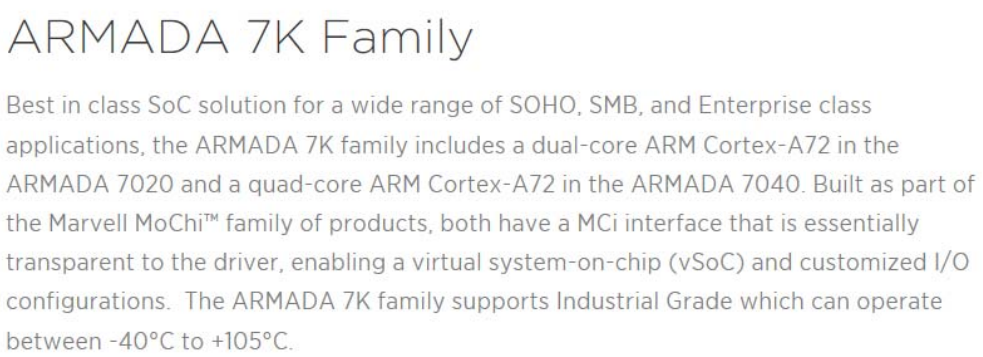
57. Cypress made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Traveo S6J3310 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products") [10]:

[10] A non-exhaustive list of additional accused products includes the Traveo S6J3320 and S6J3340 families of products that include advanced circuits for variable-length coding and decoding of media data.



Source: http://www.cypress.com/documentation/datasheets/s6j3310s6j3320s6j3330s6j3340-series-32-bit-microcontroller-traveo-family

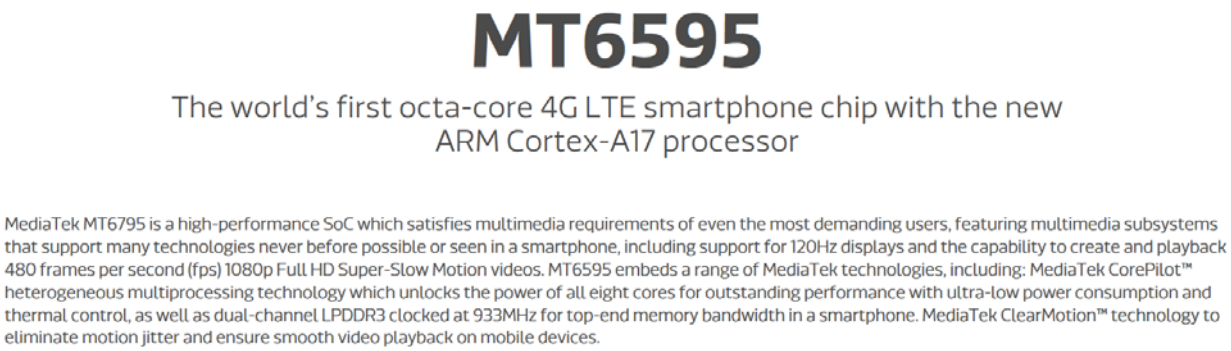
58. Marvell made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its ARMADA 7020 family of products that include advanced circuits for variable-length coding and decoding of media data ("accused products") [11]:



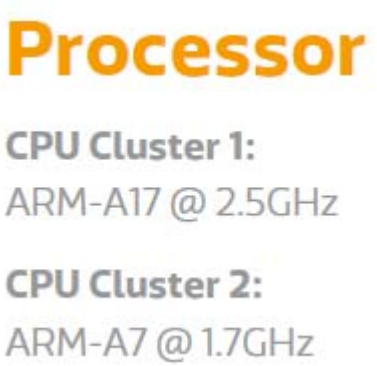
Source: https://www.marvell.com/embedded-processors/armada-70xx/

[11] A non-exhaustive list of additional accused products includes the ARMADA 7040, ARMADA 8020, and ARMADA 8040 families of products that include advanced circuits for variable-length coding and decoding of media data.

59. MediaTek made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its MT6595 and Helio X27 families of products that include advanced circuits for variable-length coding and decoding of media data ("accused products") [12]:



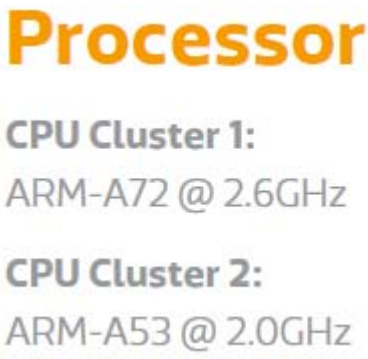
Source: https://www.mediatek.com/products/smartphones/mt6595



Source: https://www.mediatek.com/products/smartphones/mt6595

[12] A non-exhaustive list of additional accused products includes the Helio P60, Helio P70, MT8173, MT8176, and Helio X series (including Helio X20, Helio X23, and Helio X25) families of products that include advanced circuits for variable-length coding and decoding of media data.

Source: https://www.mediatek.com/products/smartphones/mt6797x-helio-x27



Source: https://www.mediatek.com/products/smartphones/mt6797x-helio-x27

60. By doing so, Defendants have directly infringed (literally and/or under the doctrine of equivalents) at least Claim 1 of the '720 Patent. Defendants' infringement in this regard is ongoing.

61. Defendants have infringed the '720 Patent by making, having made, using, importing, providing, supplying, distributing, selling or offering for sale integrated circuits having variable-length encode/decode processors.

62. The accused products include a central processing unit and a special-purpose processing unit coupled to the central processing unit and operable to process image data according to a block-based coding/decoding standard.

Media Processing Engine

The MPE implements ARM NEON technology, a media and signal processing architecture that adds instructions targeted at audio, video, 3-D graphics, image, and speech processing. Advanced SIMD instructions are available in both ARM and Thumb states.

If the design includes the MPE, the FPU is included.

See the *Cortex-A5 NEON Media Processing Engine Technical Reference Manual*.

Source: ARM Cortex-A5 Technical Reference Manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0433c/DDI0433C_cortex_a5_trm.pdf

The Cortex-A5 processor implements the ARM v7-A architecture profile. This includes:

- the 32-bit ARM instruction set
- the Thumb instruction set, a variable-length instruction set, that has both 16-bit and 32-bit instructions
- execution of 8-bit Java bytecodes
- the ThumbEE instruction set
- the security extensions, TrustZone
- the VFPv4 Floating point architecture and Advanced SIMD extensions, NEON.

Source: ARM Cortex-A5 Technical Reference Manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0433c/DDI0433C_cortex_a5_trm.pdf

ARM architecture

The Cortex-R5 processor implements the ARMv7-R architecture profile that includes the following architecture extensions:

- Advanced *Single Instruction Multiple Data* (SIMD) architecture extension for integer and floating-point vector operations
- *Vector Floating-Point version 3* (VFPv3) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard
- Multiprocessing Extensions for multiprocessing functionality.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

Note

- The Cortex-R5F processor is a Cortex-R5 processor that includes the optional *Floating Point Unit* (FPU) extension.
- In this book, references to the Cortex-R5 processor also apply to the Cortex-R5F processor, unless the context makes it clear that this is not the case.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf

Table A4-24 Miscellaneous Advanced SIMD data-processing instructions

Instruction	See
Vector Absolute Difference and Accumulate	*VABA, VABAL* on page A8-819
Vector Absolute Difference	*VABD, VABDL (integer)* on page A8-821 *VABD (floating-point)* on page A8-823
Vector Absolute	*VABS* on page A8-825
Vector Convert between floating-point and fixed point	*VCVT (between floating-point and fixed-point, Advanced SIMD)* on page A8-873
Vector Convert between floating-point and integer	*VCVT (between floating-point and integer, Advanced SIMD)* on page A8-869
Vector Convert between half-precision and single-precision	*VCVT (between half-precision and single-precision, Advanced SIMD)* on page A8-879
Vector Count Leading Sign Bits	*VCLS* on page A8-859
Vector Count Leading Zeros	*VCLZ* on page A8-863
Vector Count Set Bits	*VCNT* on page A8-867
Vector Duplicate scalar	*VDUP (scalar)* on page A8-885
Vector Extract	*VEXT* on page A8-891
Vector Move and Narrow	*VMOVN* on page A8-953
Vector Move Long	*VMOVL* on page A8-951
Vector Maximum, Minimum	*VMAX, VMIN (integer)* on page A8-927 *VMAX, VMIN (floating-point)* on page A8-929
Vector Negate	*VNEG* on page A8-969
Vector Pairwise Maximum, Minimum	*VPMAX, VPMIN (integer)* on page A8-987 *VPMAX, VPMIN (floating-point)* on page A8-989
Vector Reciprocal Estimate	*VRECPE* on page A8-1025
Vector Reciprocal Step	*VRECPS* on page A8-1027
Vector Reciprocal Square Root Estimate	*VRSQRTE* on page A8-1039
Vector Reciprocal Square Root Step	*VRSQRTS* on page A8-1041
Vector Reverse	*VREV16, VREV32, VREV64* on page A8-1029
Vector Saturating Absolute	*VQABS* on page A8-995
Vector Saturating Move and Narrow	*VQMOVN, VQMOVUN* on page A8-1005
Vector Saturating Negate	*VQNEG* on page A8-1007
Vector Swap	*VSWP* on page A8-1093
Vector Table Lookup	*VTBL, VTBX* on page A8-1095

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

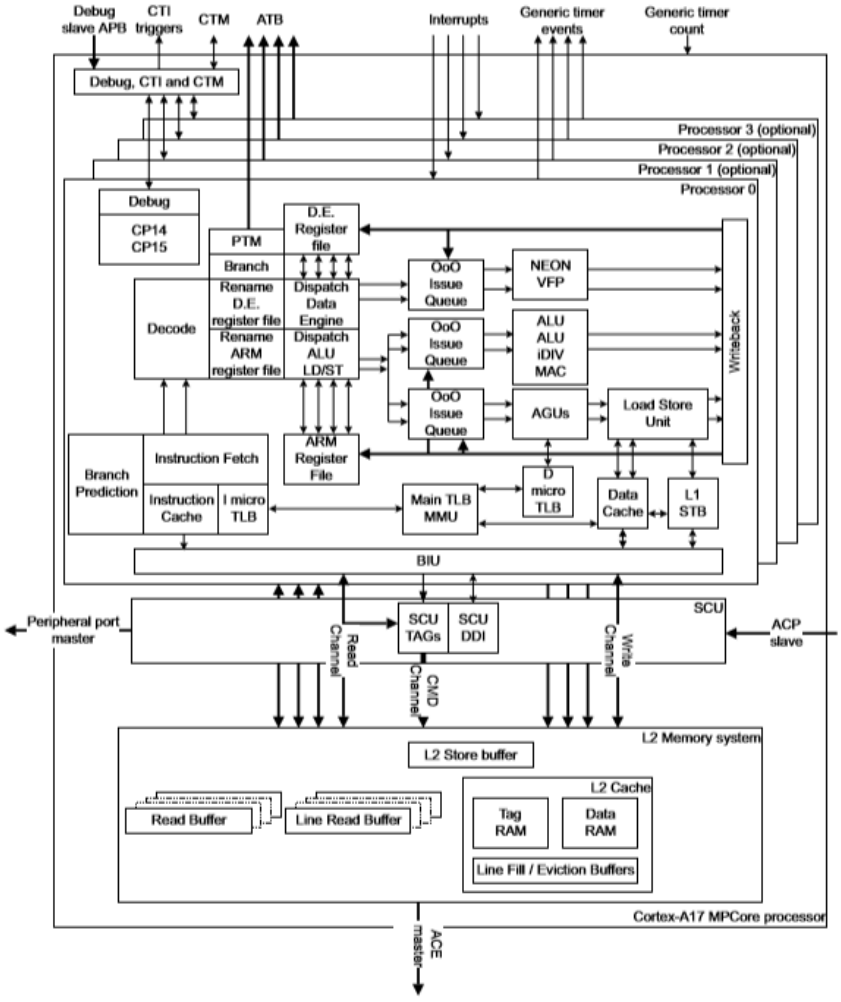
VREV16 (Vector Reverse in halfwords) reverses the order of 8-bit elements in each halfword of the vector, and places the result in the corresponding destination vector.

VREV32 (Vector Reverse in words) reverses the order of 8-bit or 16-bit elements in each word of the vector, and places the result in the corresponding destination vector.

VREV64 (Vector Reverse in doublewords) reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector, and places the result in the corresponding destination vector.

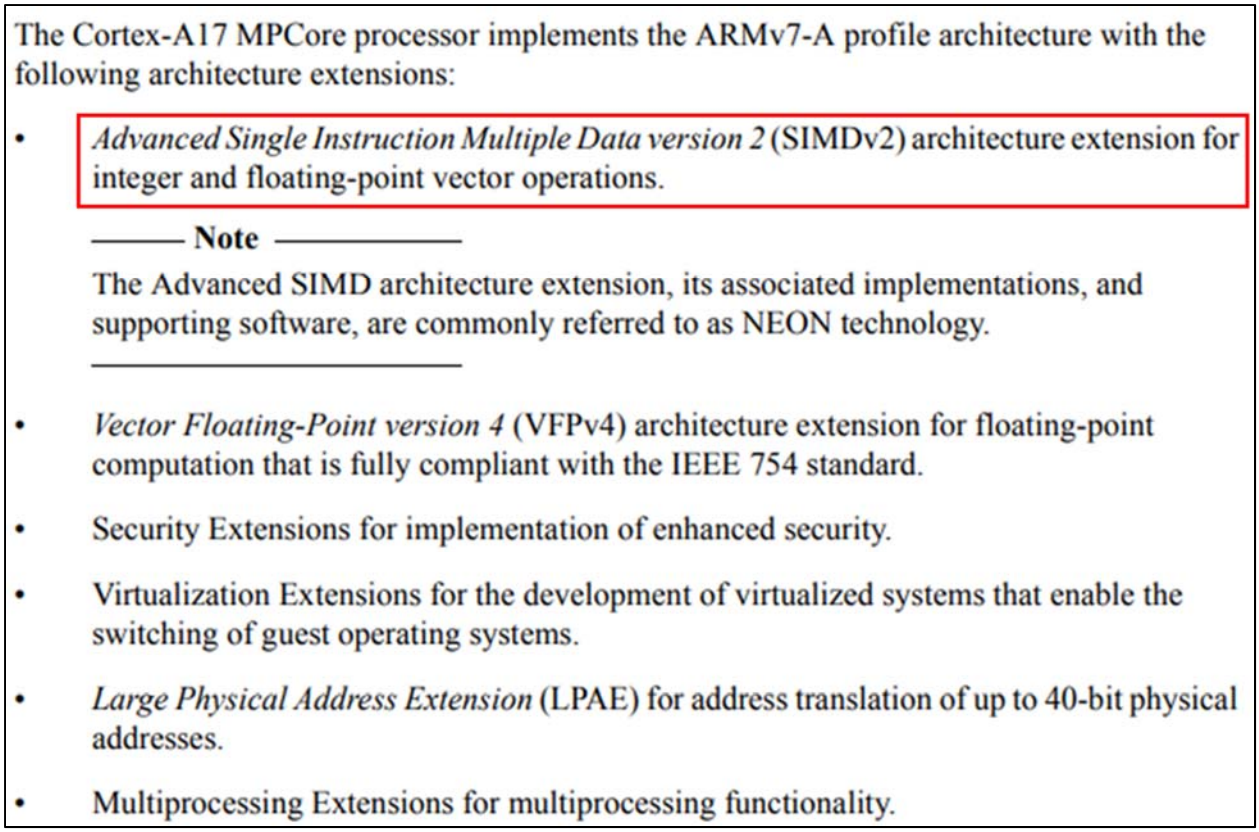
Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf



Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf



The Cortex-A17 MPCore processor implements the ARMv7-A profile architecture with the following architecture extensions:

- *Advanced Single Instruction Multiple Data version 2* (SIMDv2) architecture extension for integer and floating-point vector operations.

  Note

  The Advanced SIMD architecture extension, its associated implementations, and supporting software, are commonly referred to as NEON technology.

- *Vector Floating-Point version 4* (VFPv4) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard.
- Security Extensions for implementation of enhanced security.
- Virtualization Extensions for the development of virtualized systems that enable the switching of guest operating systems.
- *Large Physical Address Extension* (LPAE) for address translation of up to 40-bit physical addresses.
- Multiprocessing Extensions for multiprocessing functionality.

Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

Table A4-24 Miscellaneous Advanced SIMD data-processing instructions

Instruction	See
Vector Absolute Difference and Accumulate	*VABA, VABAL* on page A8-819
Vector Absolute Difference	*VABD, VABDL (integer)* on page A8-821 *VABD (floating-point)* on page A8-823
Vector Absolute	*VABS* on page A8-825
Vector Convert between floating-point and fixed point	*VCVT (between floating-point and fixed-point, Advanced SIMD)* on page A8-873
Vector Convert between floating-point and integer	*VCVT (between floating-point and integer, Advanced SIMD)* on page A8-869
Vector Convert between half-precision and single-precision	*VCVT (between half-precision and single-precision, Advanced SIMD)* on page A8-879
Vector Count Leading Sign Bits	*VCLS* on page A8-859
Vector Count Leading Zeros	*VCLZ* on page A8-863
Vector Count Set Bits	*VCNT* on page A8-867
Vector Duplicate scalar	*VDUP (scalar)* on page A8-885
Vector Extract	*VEXT* on page A8-891
Vector Move and Narrow	*VMOVN* on page A8-953
Vector Move Long	*VMOVL* on page A8-951
Vector Maximum, Minimum	*VMAX, VMIN (integer)* on page A8-927 *VMAX, VMIN (floating-point)* on page A8-929
Vector Negate	*VNEG* on page A8-969
Vector Pairwise Maximum, Minimum	*VPMAX, VPMIN (integer)* on page A8-987 *VPMAX, VPMIN (floating-point)* on page A8-989
Vector Reciprocal Estimate	*VRECPE* on page A8-1025
Vector Reciprocal Step	*VRECPS* on page A8-1027
Vector Reciprocal Square Root Estimate	*VRSQRTE* on page A8-1039
Vector Reciprocal Square Root Step	*VRSQRTS* on page A8-1041
Vector Reverse	*VREV16, VREV32, VREV64* on page A8-1029
Vector Saturating Absolute	*VQABS* on page A8-995
Vector Saturating Move and Narrow	*VQMOVN, VQMOVUN* on page A8-1005
Vector Saturating Negate	*VQNEG* on page A8-1007
Vector Swap	*VSWP* on page A8-1093
Vector Table Lookup	*VTBL, VTBX* on page A8-1095

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

VREV16, VREV32, VREV64

VREV16 (Vector Reverse in halfwords) reverses the order of 8-bit elements in each halfword of the vector, and places the result in the corresponding destination vector.

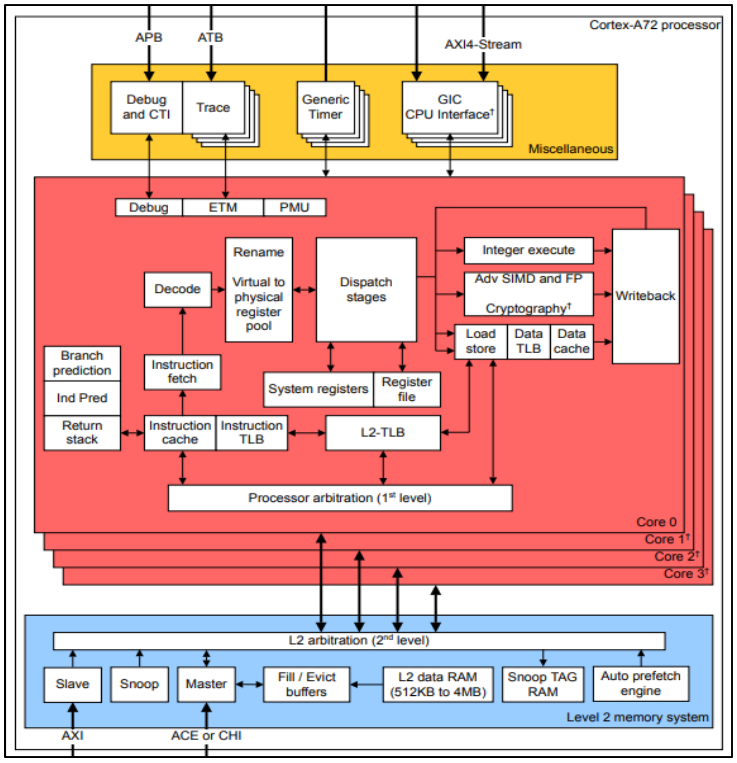
VREV32 (Vector Reverse in words) reverses the order of 8-bit or 16-bit elements in each word of the vector, and places the result in the corresponding destination vector.

VREV64 (Vector Reverse in doublewords) reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector, and places the result in the corresponding destination vector.

There is no distinction between data types, other than size.

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf



Source: ARM Cortex-A72 MPCore Processor manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.100095_0003_06_en/cortex_a72_mpcore_trm_100095_0003_06_en.pdf

Advanced SIMD and Floating-point unit

The Advanced SIMD and Floating-point unit provides support for the ARMv8 Advanced SIMD and Floating-point execution. In addition, the Advanced SIMD and Floating-point unit provides support for the optional Cryptography engine.

Source: ARM Cortex-A72 MPCore Processor manual downloaded from

http://infocenter.arm.com/help/topic/com.arm.doc.100095_0003_06_en/cortex_a72_mpcore_trm_100095_0003_06_en.pdf

Table B2-2 Byte reversal instructions

Function	Instructions	Notes
Reverse bytes in 32-bit word or words[a]	REV32	For use with general-purpose registers
Reverse bytes in whole register	REV	For use with general-purpose registers
Reverse bytes in 16-bit halfwords	REV16	For use with general-purpose registers
Reverse elements in doublewords, vector	REV64	For use with SIMD and floating-point registers
Reverse elements in words, vector	REV32	For use with SIMD and floating-point registers
Reverse elements in halfwords, vector	REV16	For use with SIMD and floating-point registers

a. Can operate on multiple words.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

REV64

Reverse elements in 64-bit doublewords (vector). This instruction reverses the order of 8-bit, 16-bit, or 32-bit elements in each doubleword of the vector in the source SIMD&FP register, places the results into a vector, and writes the vector to the destination SIMD&FP register.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

NEON

Arm NEON technology is an advanced SIMD (single instruction multiple data) architecture extension for the Arm Cortex-A series and Cortex-R52 processors.

Source: https://developer.arm.com/technologies/neon

Examples of some key available functions are detailed below:

Video codecs:	Audio codecs:	Voice and speech codecs:	Audio enhancement algorithms:	Computer Vision	Machine and deep learning
VP9 OTT encoder, VP9 Consumer encoder/decoder	MP3 encoder/decoder	G.711	Echo cancellation	Canny Edge detection	On-device object recognition
H.264 (AVC) encoder/decoder	MPEG-2 layer I & II encoder/decoder	G.722, G.722.1, G.722.2	Noise Reduction	Harris Corner	On-device scene recognition
MPEG4 SP/ASP encoder/decoder	MPEG-1 layer III audio encoder	G.723.1	Beam Forming	ORB	Human pose recognition
MPEG2 decoder	MPEG-1 layer III audio encoder/decoder	G.726	Comfort Noise	Convolution filter	Defect detection
H.263 decoder	HE-AACv1, v2 encoder/decoder	G.727	AudioZoom	Erosion/Dilation	

Source: https://developer.arm.com/technologies/neon

63. The accused products include an input/output buffer coupled to the special-purpose processing unit and operable to transfer the image data between the special-purpose processing unit and an input/output bus.

64. The special-purpose processing unit of the accused products is operable to address a memory location with a value stored in a symbol register, retrieve an address from the addressed location, and store the retrieved address in the symbol register.

About the Cortex-A5 NEON MPE

The Cortex-A5 NEON MPE extends the Cortex-A5 functionality to provide support for the ARM v7 Advanced SIMD v2 and *Vector Floating-Point v4* (VFPv4) instruction sets.

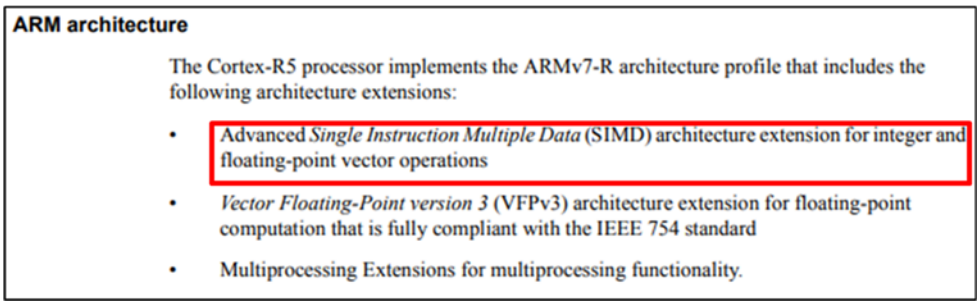
The NEON MPE supports all addressing modes and operations that are described in the *ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition*.

The NEON MPE features are:

- SIMD and scalar single-precision floating-point computation.
- Scalar double-precision floating-point computation.
- SIMD and scalar half-precision floating-point conversion.
- SIMD 8, 16, 32, and 64-bit signed and unsigned integer computation.
- 8 or 16-bit polynomial computation for single-bit coefficients.
- Structured data load capabilities.
- Large, shared register file, addressable as:
  - Thirty-two 32-bit S (single) registers.
  - Thirty-two 64-bit D (double) registers.
  - Sixteen 128-bit Q (quad) registers See the *ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition* for details of the extension register set.

Source: ARM Cortex-A5 NEON Media Processing Engine Technical Reference Manual downloaded from

https://static.docs.arm.com/100304/0001/cortex_a5_neon_mpe_trm_100304_0001_00_en.pdf



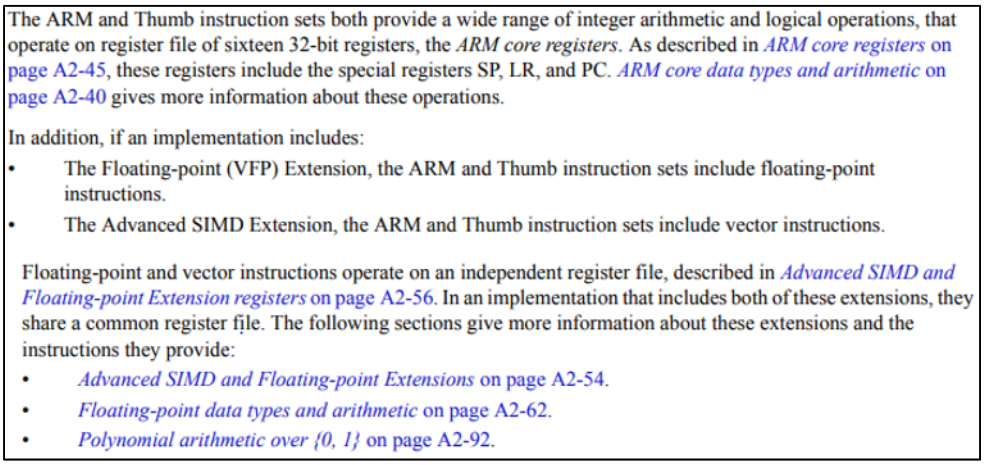
ARM architecture

The Cortex-R5 processor implements the ARMv7-R architecture profile that includes the following architecture extensions:

- Advanced *Single Instruction Multiple Data* (SIMD) architecture extension for integer and floating-point vector operations
- *Vector Floating-Point version 3* (VFPv3) architecture extension for floating-point computation that is fully compliant with the IEEE 754 standard
- Multiprocessing Extensions for multiprocessing functionality.

Source: ARM Cortex R5 Reference Manual downloaded from

https://static.docs.arm.com/ddi0460/d/DDI0460D_cortex_r5_r1p2_trm.pdf



The ARM and Thumb instruction sets both provide a wide range of integer arithmetic and logical operations, that operate on register file of sixteen 32-bit registers, the *ARM core registers*. As described in *ARM core registers* on page A2-45, these registers include the special registers SP, LR, and PC. *ARM core data types and arithmetic* on page A2-40 gives more information about these operations.

In addition, if an implementation includes:

- The Floating-point (VFP) Extension, the ARM and Thumb instruction sets include floating-point instructions.
- The Advanced SIMD Extension, the ARM and Thumb instruction sets include vector instructions.

Floating-point and vector instructions operate on an independent register file, described in *Advanced SIMD and Floating-point Extension registers* on page A2-56. In an implementation that includes both of these extensions, they share a common register file. The following sections give more information about these extensions and the instructions they provide:

- *Advanced SIMD and Floating-point Extensions* on page A2-54.
- *Floating-point data types and arithmetic* on page A2-62.
- *Polynomial arithmetic over {0, 1}* on page A2-92.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

From VFPv3, the Advanced SIMD and Floating-point (VFP) Extensions use the same register set. This is distinct from the ARM core register set. These registers are generally referred to as the *extension registers*.

The extension register set consists of either 32 or 16 doubleword registers, as follows:

- If VFPv2 is implemented, it consists of 16 doubleword registers.
- If VFPv3 is implemented, it consists of either 32 or 16 doubleword registers. Where necessary, these two implementation options are distinguished using the terms:
  - VFPv3-D32, for an implementation with 32 doubleword registers.
  - VFPv3-D16, for an implementation with 16 doubleword registers.
- If VFPv4 is implemented, it consists of either 32 or 16 doubleword registers. Where necessary, these two implementation options are distinguished using the terms:
  - VFPv4-D32, for an implementation with 32 doubleword registers.
  - VFPv4-D16, for an implementation with 16 doubleword registers.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

A4.11 Advanced SIMD and Floating-point load/store instructions

Table A4-16 summarizes the extension register load/store instructions in the Advanced SIMD and Floating-point (VFP) instruction sets.

Advanced SIMD also provides instructions for loading and storing multiple elements, or structures of elements, see *Element and structure load/store instructions*.

Table A4-16 Extension register load/store instructions

Instruction	See	Operation
Vector Load Multiple	*VLDM* on page A8-923	Load 1-16 consecutive 64-bit registers, Advanced SIMD and Floating-point Load 1-16 consecutive 32-bit registers, Floating-point only
Vector Load Register	*VLDR* on page A8-925	Load one 64-bit register, Advanced SIMD and Floating-point Load one 32-bit register, Floating-point only
Vector Store Multiple	*VSTM* on page A8-1081	Store 1-16 consecutive 64-bit registers, Advanced SIMD and Floating-point Store 1-16 consecutive 32-bit registers, Floating-point only
Vector Store Register	*VSTR* on page A8-1083	Store one 64-bit register, Advanced SIMD and Floating-point Store one 32-bit register, Floating-point only

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

VLDR

This instruction loads a single extension register from memory, using an address from an ARM core register, with an optional offset.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

VSTR

This instruction stores a single extension register to memory, using an address from an ARM core register, with an optional offset.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

VTBL, VTBX

Vector Table Lookup uses byte indexes in a control vector to look up byte values in a table and generate a new vector. Indexes out of range return 0.

Vector Table Extension works in the same way, except that indexes out of range leave the destination element unchanged.

Depending on settings in the CPACR, NSACR, and HCPTR registers, and the security state and mode in which the instruction is executed, an attempt to execute the instruction might be UNDEFINED, or trapped to Hyp mode. *Summary of access controls for Advanced SIMD functionality* on page B1-1232 summarizes these controls.

ARM deprecates the conditional execution of any Advanced SIMD instruction encoding that is not also available as a VFP instruction encoding, see *Conditional execution* on page A8-286.

Source: ARMv7-R Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

Instruction sets, arithmetic operations, and register files

The ARM and Thumb instruction sets both provide a wide range of integer arithmetic and logical operations, that operate on register file of sixteen 32-bit registers, the *ARM core registers*. As described in *ARM core registers* on page A2-45, these registers include the special registers SP, LR, and PC. *ARM core data types and arithmetic* on page A2-40 gives more information about these operations.

In addition, if an implementation includes:

- The Floating-point (VFP) Extension, the ARM and Thumb instruction sets include floating-point instructions.
- The Advanced SIMD Extension, the ARM and Thumb instruction sets include vector instructions.

Floating-point and vector instructions operate on an independent register file, described in *Advanced SIMD and Floating-point Extension registers* on page A2-56. In an implementation that includes both of these extensions, they share a common register file. The following sections give more information about these extensions and the instructions they provide:

- *Advanced SIMD and Floating-point Extensions* on page A2-54.
- *Floating-point data types and arithmetic* on page A2-62.
- *Polynomial arithmetic over {0, 1}* on page A2-92.

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

> The processor supports all addressing modes, data types, and operations in the VFPv4 extension with version 3 of the Common VFP subarchitecture. The processor implements VFPv4-D32. See the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* for information on the VFPv4 instruction set.

Source: ARM Cortex-A17 MPCore Processor manual downloaded from

https://static.docs.arm.com/ddi0535/c/DDI0535C_cortex_a17_r1p1_trm.pdf

> From VFPv3, the Advanced SIMD and Floating-point (VFP) Extensions use the same register set. This is distinct from the ARM core register set. These registers are generally referred to as the *extension registers*.
>
> The extension register set consists of either 32 or 16 doubleword registers, as follows:
>
> - If VFPv2 is implemented, it consists of 16 doubleword registers.
> - If VFPv3 is implemented, it consists of either 32 or 16 doubleword registers. Where necessary, these two implementation options are distinguished using the terms:
>   - VFPv3-D32, for an implementation with 32 doubleword registers.
>   - VFPv3-D16, for an implementation with 16 doubleword registers.
> - If VFPv4 is implemented, it consists of either 32 or 16 doubleword registers. Where necessary, these two implementation options are distinguished using the terms:
>   - VFPv4-D32, for an implementation with 32 doubleword registers.
>   - VFPv4-D16, for an implementation with 16 doubleword registers.

Source: ARMv7-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0406/cd/DDI0406C_d_armv7ar_arm.pdf

> The ARMv8 architecture provides two register files:
> - A general-purpose register file.
> - A SIMD&FP register file.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

Vector formats

In an implementation that includes the SIMD instructions that operate on the SIMD&FP register file, a register can hold one or more packed elements, all of the same size and type. The combination of a register and a data type describes a vector of elements. The vector is considered to be an array of elements of the data type specified in the instruction. The number of elements in the vector is implied by the size of the data elements and the size of the register.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

TBL

Table vector Lookup. This instruction reads each value from the vector elements in the index source SIMD&FP register, uses each result as an index to perform a lookup in a table of bytes that is described by one to four source table SIMD&FP registers, places the lookup result in a vector, and writes the vector to the destination SIMD&FP register. If an index is out of range for the table, the result for that lookup is 0. If more than one source register is used to describe the table, the first source register describes the lowest bytes of the table.

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

SIMD vector register list

Where an instruction operates on multiple SIMD and floating-point registers, for example vector Load/Store structure and table lookup operations, the registers are specified as a list enclosed by curly braces. This list consists of either a sequence of registers separated by commas, or a register range separated by a hyphen. The registers must be numbered in increasing order, modulo 32, in increments of one. The hyphenated form is preferred for disassembly if there are more than two registers in the list and the register number are increasing. The following examples are equivalent representations of a set of four registers `V4` to `V7`, each holding four lanes of 32-bit elements:

Source: ARMv8-A Architecture Reference Manual downloaded from

https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf?_ga=2.33769697.1581472737.1542011475-1643828834.1539593502

65. Defendants have had knowledge of the ‘720 Patent at least as of the date when they were notified of the filing of this action.

66. American Patents has been damaged as a result of the infringing conduct by Defendants alleged above. Thus, Defendants are liable to American Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

67. American Patents and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the '720 Patent.

ADDITIONAL ALLEGATIONS REGARDING INDIRECT INFRINGEMENT

68. Defendants have also indirectly infringed the '293 Patent, the '058 Patent, and the '720 Patent by inducing others to directly infringe the '293 Patent, the '058 Patent, and the '720 Patent. Defendants have induced the end-users, its customers, to directly infringe (literally and/or under the doctrine of equivalents) the '293 Patent, the '058 Patent, and the '720 Patent by using the accused products. Defendants took active steps, directly and/or through contractual relationships with others, with the specific intent to cause them to use the accused products in a manner that infringes one or more claims of the patents-in-suit, including, for example, Claim 7 of the '293 Patent, Claim 1 of the '058 Patent, and Claim 1 of the '720 Patent. Such steps by Defendants included, among other things, advising or directing customers and end-users to use the accused products in an infringing manner; advertising and promoting the use of the accused products in an infringing manner; and/or distributing instructions that guide users to use the accused products in an infringing manner. Defendants are performing these steps, which constitute induced infringement, with the knowledge of the '293 Patent, the '058 Patent, and the '720 Patent and with the knowledge that the induced acts constitute infringement. Defendants were and are aware that the normal and customary use of the accused products by Defendants'

customers would infringe the '293 Patent, the '058 Patent, and the '720 Patent. Defendants' inducement is ongoing.

69. Defendants have also induced its affiliates, or third-party manufacturers, shippers, distributors, retailers, or other persons acting on its or its affiliates' behalf, to directly infringe (literally and/or under the doctrine of equivalents) the '293 Patent, the '058 Patent, and the '720 Patent by importing, selling or offering to sell the accused products. Defendants took active steps, directly and/or through contractual relationships with others, with the specific intent to cause such persons to import, sell, or offer to sell the accused products in a manner that infringes one or more claims of the patents-in-suit, including, for example, Claim 7 of the '293 Patent, Claim 1 of the '058 Patent, and Claim 1 of the '720 Patent. Such steps by Defendants included, among other things, making or selling the accused products outside of the United States for importation into or sale in the United States, or knowing that such importation or sale would occur; and directing, facilitating, or influencing its affiliates, or third-party manufacturers, shippers, distributors, retailers, or other persons acting on its behalf, to import, sell, or offer to sell the accused products in an infringing manner. Defendants performed these steps, which constitute induced infringement, with the knowledge of the '293 Patent, the '058 Patent, and the '720 Patent and with the knowledge that the induced acts would constitute infringement. Defendants performed such steps in order to profit from the eventual sale of the accused products in the United States. Defendants' inducement is ongoing.

70. Defendants have also indirectly infringed by contributing to the infringement of the '293 Patent, the '058 Patent, and the '720 Patent. Defendants have contributed to the direct infringement of the '293 Patent, the '058 Patent, and the '720 Patent by the end-user of the accused products. The accused products have special features that are specially designed to be

used in an infringing way and that have no substantial uses other than ones that infringe the '293 Patent, the '058 Patent, and the '720 Patent, including, for example, Claim 7 of the '293 Patent, Claim 1 of the '058 Patent, and Claim 1 of the '720 Patent. The special features include advanced circuits for variable-length coding and decoding of media data in a manner that infringes the '293 Patent, the '058 Patent, and the '720 Patent. The special features constitute a material part of the invention of one or more of the claims of the '293 Patent, the '058 Patent, and the '720 Patent and are not staple articles of commerce suitable for substantial non-infringing use. Defendants' contributory infringement is ongoing.

71. Furthermore, Defendants have a policy or practice of not reviewing the patents of others (including instructing its employees to not review the patents of others), and thus has been willfully blind of American Patents' patent rights.

72. Defendants' actions are at least objectively reckless as to the risk of infringing valid patents and this objective risk was either known or should have been known by Defendants.

73. Defendants' direct and indirect infringement of the '293 Patent, the '058 Patent, and the '720 Patent is, has been, and continues to be willful, intentional, deliberate, and/or in conscious disregard of American Patents' rights under the patents.

74. American Patents has been damaged as a result of the infringing conduct by Defendants alleged above. Thus, Defendants are liable to American Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

JURY DEMAND

American Patents hereby requests a trial by jury on all issues so triable by right.

PRAYER FOR RELIEF

American Patents requests that the Court find in its favor and against Defendants, and that the Court grant American Patents the following relief:

a. Judgment that one or more claims of the '293 Patent, the '058 Patent, and the '720 Patent have been infringed, either literally and/or under the doctrine of equivalents, by Defendants and/or all others acting in concert therewith;

b. A permanent injunction enjoining Defendants and their officers, directors, agents, servants, affiliates, employees, divisions, branches, subsidiaries, parents, and all others acting in concert therewith from infringement of the '293 Patent, the '058 Patent, and the '720 Patent; or, in the alternative, an award of a reasonable ongoing royalty for future infringement of the '293 Patent, the '058 Patent, and the '720 Patent;

e. Judgment that Defendants account for and pay to American Patents all damages to and costs incurred by American Patents because of Defendants' infringing activities and other conduct complained of herein, including an award of all increased damages to which American Patents is entitled under 35 U.S.C. § 284;

f. That American Patents be granted pre-judgment and post-judgment interest on the damages caused by Defendants' infringing activities and other conduct complained of herein;

g. That this Court declare this an exceptional case and award American Patents its reasonable attorney's fees and costs in accordance with 35 U.S.C. § 285; and

h. That American Patents be granted such other and further relief as the Court may deem just and proper under the circumstances.

Dated: December 6, 2018

Respectfully submitted,

/s/ *Stafford Davis*
Stafford Davis
State Bar No. 24054605
sdavis@stafforddavisfirm.com

Catherine Bartles (*admission pending*)
Texas Bar No. 24104849
cbartles@stafforddavisfirm.com
THE STAFFORD DAVIS FIRM
The People's Petroleum Building
102 North College Avenue, 13th Floor
Tyler, Texas 75702
(903) 593-7000
(903) 705-7369 fax

Matthew J. Antonelli (*admission pending*)
Texas Bar No. 24068432
matt@ahtlawfirm.com
Zachariah S. Harrington (*admission pending*)
Texas Bar No. 24057886
zac@ahtlawfirm.com
Larry D. Thompson, Jr. (*admission pending*)
Texas Bar No. 24051428
larry@ahtlawfirm.com
Christopher Ryan Pinckney (*admission pending*)
Texas Bar No. 24067819
ryan@ahtlawfirm.com
Michael D. Ellis
Texas Bar No. 24081586
michael@ahtlawfirm.com

ANTONELLI, HARRINGTON
& THOMPSON LLP
4306 Yoakum Blvd., Ste. 450
Houston, TX 77006
(713) 581-3000

Attorneys for American Patents LLC